Statistical identification of faults in 3D seismic volumes using a machine learning approach

Antoine Guitton[1], Hua Wang[2], and Whitney Trainor-Guitton[1]
[1] *Geophysics Department, Colorado School of Mines, Golden, Colorado*
[2] *Computer Science Department, Colorado School of Mines, Golden, Colorado*

ABSTRACT
Faults are highlighted in 3D seismic volumes with a supervised machine learning algorithm. We label the faults using an automatic fault-picking method developed by Hale (2013). We build feature vectors for the training and classification steps using two popular techniques in object recognition algorithms called Histograms of Oriented Gradients (HOG) and Scale Invariant Feature Transforms (SIFT). We train and classify the seismic data using a Support Vector Machine classifier with Gaussian kernels. Using both SIFT and HOG features together reduces the false positive rate, thus delivering better fault images. Our approach is able to predict faults in both synthetic and field data cubes quite well even when mislabeled data are used for training. We propose many promising research directions to improve on our approach.

Key words: Faults, machine learning, statistics, HOG, SIFT, SVM

1 INTRODUCTION

Interpretation tasks requiring human input are by nature slow, expensive, and not repeatable. In exploration geophysics, the interpretation of large seismic volumes (tens or hundreds of gigabytes) remains one of the most labor-intensive task. Enormous progress has been achieved to facilitate these tedious steps. The automatic interpretation of 3D volumes based on semblance analysis, phase unwrapping (Stark, 2004) or local dip integration (Lomask et al., 2006) has revolutionized the way interpretation can be done: using deterministic approaches and simple differential operators (Hale, 2013), codes can reliably extract the information of interest in large volumes. Contrary to humans, interpretation tasks done by computers are fast, cheap and repeatable. In addition, whereas humans are particularly efficient at finding features in 2D only, properly written softwares can work in any dimension.

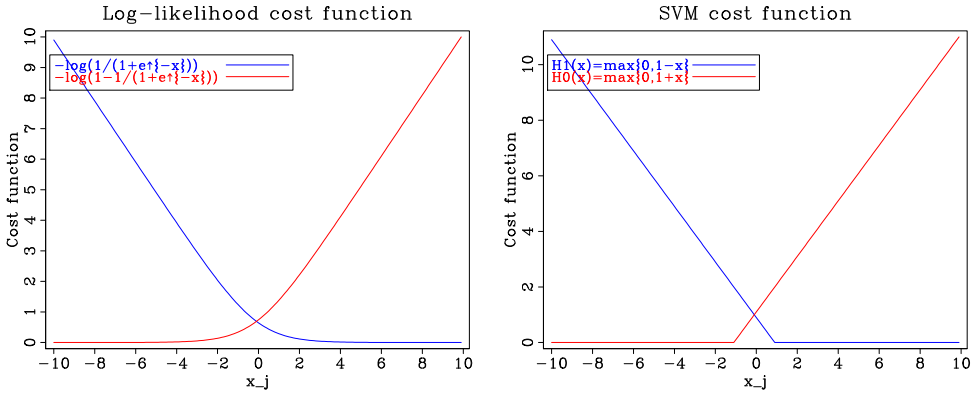
In this paper, we propose a fault highlighting technique that follows in the footsteps of a long tradition of automated seismic interpretation methods. Marfurt et al. (1998) extract seismic attributes using a semblance-based coherency algorithm. This method looks at measuring the continuity of seismic events to detect faults, for instance. Other measures of continuity using variance of gradient magnitude are also possible. Other authors have looked at picking fault surfaces automatically using, for instance, ant tracking (Pedersen et al., 2005).

Here, we take a statistical approach to the interpretation of faults and use machine learning algorithms (MLAs) to identify them. Our goal is not to pick fault surfaces, but to provide maps of possible fault locations (also known as fault imaging or highlighting). Our method follows a supervised learning concept. In a nutshell, for input, we have labeled seismic sections where faults are picked and unlabeled seismic sections where faults are unknown. From the labeled seismic data, we extract a set of features (that we define later) that will be used to train the MLA to identify faults. Once the MLA is trained, it can predict fault locations on the unlabeled data. A similar approach is used by Araya-Polo et al. (2017) using features estimated in the prestack domain and Huang et al. (2017) using features computed by the fault detection method of Hale (2013). Both use neural networks for the classification step.

We first start our manuscript by presenting basic MLA concepts and notations. For the fault imaging problem, we show how to obtain our labeled data, how to extract features, and how to use them to train the MLA. We illustrate our method with 3D synthetic and field datasets. In both cases, the MLA is able to identify faults very well.

2 CONCEPTS OF MACHINE LEARNING

In this section, we introduce basic notations and concepts to better understand machine learning algorithms. Machine learning can be applied to labeled, semi-labeled, or unlabeled data. In the labeled case, all samples used for the training have a quantitative or qualitative value. For instance, to the question "what is on this photo", we can assign to pictures qualitative



(a) (b)

Figure 1. Cost functions for (a) logistic regression and (b) SVM classifier.

values such as "truck", "car", "person", "house", etc... To the question "what is the estimated selling price of my home", we can assign a quantitative value, like "$300,000". Because labeling can be quite cumbersome or impossible due to the volume of data to process, we might label only few data points, and use a semi-supervised approach, or label no data points at all, and use an unsupervised approach. In the last case, clustering techniques are quite popular to automatically label data based on their proximity to centroids. These techniques suffer from the curse of dimensionality, however, and other methods based on convolutional neural-networks might be preferred.

For our fault highlighting method, we follow a supervised learning approach. Our outcome can take only two qualitative values, or classes: $y = \{\texttt{fault,no fault}\}$. For convenience, we prefer working with values $y = \{1, 0\}$, where $y = 1$ means a fault is present, and $y = 0$ means no fault is present. Now from the seismic data, we can extract at each point some features. These features can be, for instance, gradients of the image, or semblances, or simply pixel values. A data point can be a collection of pixels in a small neighborhood, or patch. So at each point, we have a set of features that we can arrange in a vector form. We write x_{ij} the value of the feature i for data point j. For each point, we have a feature vector $\mathbf{x}_j$ and we want to find a function $f(\mathbf{x}_j)$ such that

$$y_j = f(\mathbf{x}_j) \tag{1}$$

Putting all the feature vectors into one matrix $\mathbf{X}$ and all outcomes in one column vector $\mathbf{y}$ we seek a function $f(\mathbf{X})$ such that

$$\mathbf{y} = f(\mathbf{X}) \tag{2}$$

2.1 Linear regression model

In supervised learning, we know $\mathbf{y}$ and $\mathbf{x}$ and are seeking the best function $f(\mathbf{x})$, also called a predictor or hypothesis. One of the simplest function we can think of is linear in its arguments:

$$f_r(\mathbf{x}_j) = \theta^T \mathbf{x}_j \tag{3}$$

where the unknown coefficients vector θ can be estimated minimizing, for instance, the ℓ^2 norm of the prediction misfit:

$$g_r(\theta) = \|\mathbf{y} - \mathbf{X}\theta\|_2^2. \tag{4}$$

This MLA is merely linear regression: it makes strong assumptions about the relationship between the features (linear) and works only for quantitative outcomes (y is a value). In machine learning jargon, linear regression is said to have a strong bias. To stabilize the estimates of θ, a regularization term is often added:

$$g_r(\theta) = \|\mathbf{y} - \mathbf{X}\theta\|_2^2 + \lambda\|\theta\|_2^2 \tag{5}$$

which any geophysicist will recognize as Tikhonov regularization. In MLA jargon, this is called ridge regression.

2.2 Logistic regression

To accommodate qualitative outcomes, a logistic function can be added in the definition of $f(\mathbf{x}_j)$:

$$f_l(\mathbf{x}_j) = \frac{1}{1 + e^{-\theta^T \mathbf{x}_j}} \tag{6}$$

and the outcome depends on whether $f_l(\mathbf{x}_j)$ is greater or less than 0.

$$\begin{cases} y_j = 1 & \text{if } f_l(\mathbf{x}_j) > 0 \\ y_j = 0 & \text{otherwise} \end{cases} \tag{7}$$

Logistic regression also gives us the class probabilities. If the outcome vector is $y = \{0, 1\}$, a slightly different cost function invoking the log-likelihood is minimized (Figure 1(a)):

$$g_l(\theta) = -\sum_j y_j \log(f_l(\mathbf{x}_j)) + (1 - y_j)\log(1 - f_l(\mathbf{x}_j)) + \lambda\|\theta\|_2^2 \tag{8}$$

Note that when y_j is equal to one, the second term in the sum is zero, while the first term is zero when y_j is equal to zero. Logistic and linear regressions are some of the simplest algorithms available for machine learning and are quite popular.

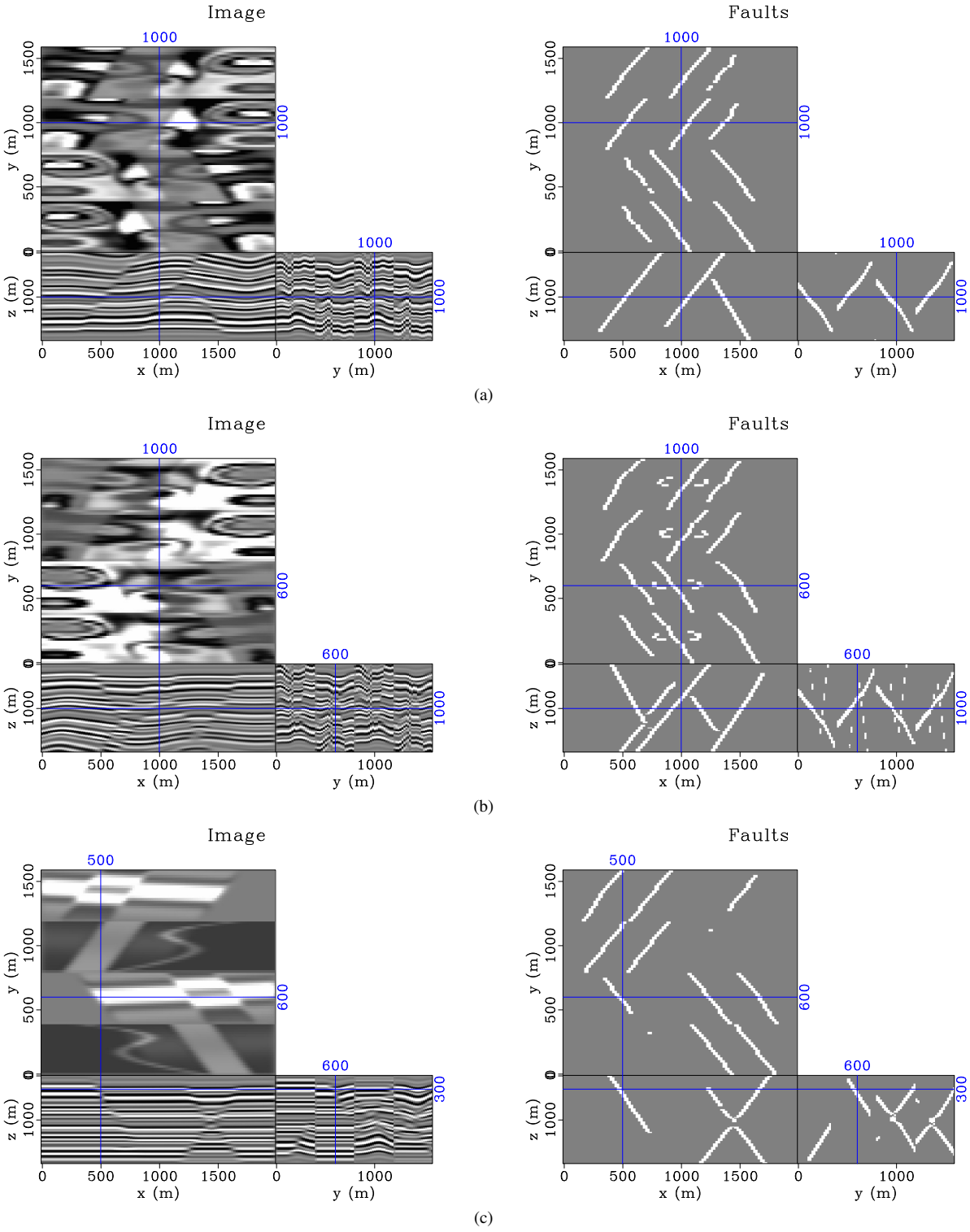


Figure 2. Synthetic seismic data volumes with their interpreted faults. Conical faults are present in (b) and can be seen as round shapes in the depth slices. Conjugate faults are present in (c) and can be seen as crosses in the inline/crossline slices. These three volumes are used for training only.

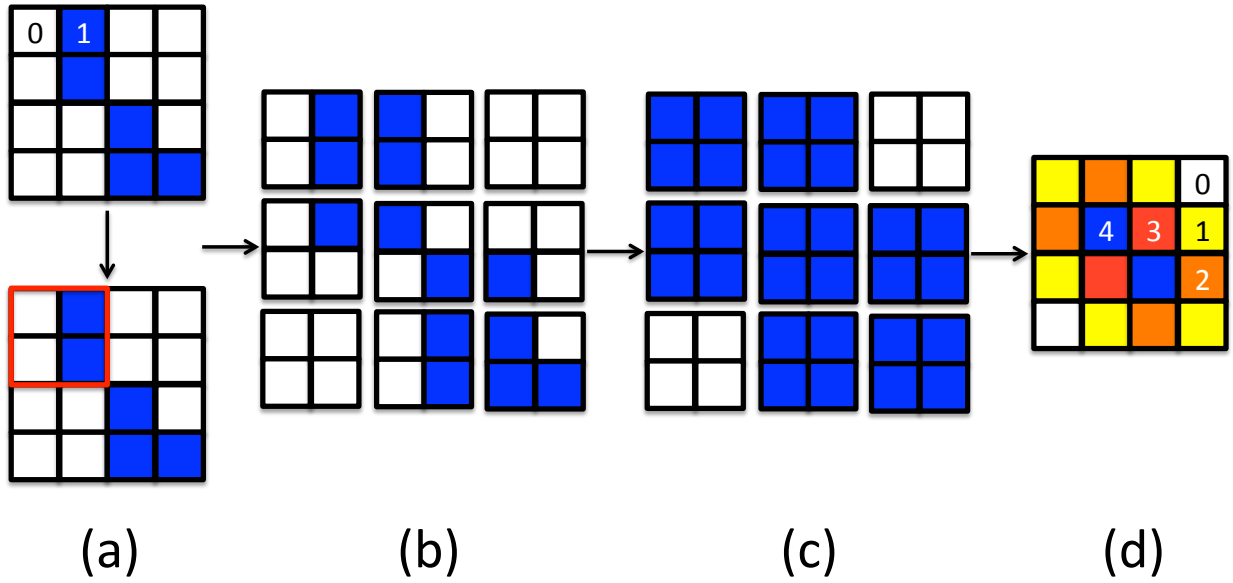


Figure 3. Patching sequence for the outcome vector **y**. (a) Fault mask equal to 1 or 0 according to the presence of a fault, or not. (b) Patching outcome for 2x2 cells. (c) The whole patch takes the value 1 if at least one cell is equal to 1, thus building the target vector **y**. (d) Reassembling the patches in (c) back together produces a smooth version of the fault mask in (a). The numbers in (d) are then rescaled to fall within the original [0,1] range.

The definitions in equations 3 and 6 can be extended to include non-linear relationships between features and/or power functions of the features themselves. In practice, however, other predictors are often preferred to so. Therefore, for fault imaging where such non-linear effects might be present, we use a support-vector machine (SVM) classifier instead. SVM classifiers are computationally efficient and work very well for non-linearly separable classes. We present a short description of SVMs below.

2.3 Support vector machine classifier

Support vector machine (SVM) classifiers are very popular to build an optimal hyperplane separating two classes (Hastie et al., 2001). It can also discriminate between non-linearly separable classes using so-called "kernels", or similarity functions. It is not the goal of this paper to describe the SVM classifier in details, but merely expose its interesting properties.

SVM classifier are called wide-margin classifiers, meaning that they find the boundary between two classes that maximize the distance between feature vectors and decision boundaries. The vectors defining the position of the margin (and on the margin) are called the support vectors.

The penalty, or loss, function associated with the SVM classifier is very similar to the loss function of equation (8) for logistic regression (Figure 1(b)) and takes the form

$$g_s(\theta) = \sum_j y_j H_1(f_s(\mathbf{x}_j)) + (1-y_j) H_0(1-f_s(\mathbf{x}_j)) + \lambda \|\theta\|_2^2 \tag{9}$$

where H_1 and H_0 are hinge loss functions defined as

$$\begin{array}{rcl} H_1(x) & = & \max\{0, 1-x\} \\ H_0(x) & = & \max\{0, 1+x\} \end{array} \tag{10}$$

With this definition, points within a class have a weight of zero and do not contribute to the total loss. Only points on the margin or between the two margins count. In addition, the contribution of these points is linear adding robustness to outliers.

To accommodate non-linearly separable classes, the notion of kernels enters into the definition of the hypothesis f_s as follows (James et al., 2013)

$$f_s(\mathbf{x}_j) = \theta^T h(\mathbf{x}_j) \tag{11}$$

and

$$h(\mathbf{x}_j) = \sum_{i=1}^{N} K(\mathbf{x}_j, \mathbf{x}_i) \tag{12}$$

where $K(\mathbf{x}, \mathbf{x}')$ is called the kernel and should be a positive semi-definite function and N is the number of training points. Kernels help defining a metric relating two vectors. They are also called similarity functions and create a local weight around each point. In this paper, we use the radial/Gaussian basis function defined as

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2). \tag{13}$$

The classification is done by estimating f_s in equation 11:

$$\begin{cases} y_j = 1 & \text{if } f_s(\mathbf{x}_j) > 0 \\ y_j = 0 & \text{otherwise} \end{cases} \tag{14}$$

Because of the shape of the cost function the SVM classifier,

contrary to logistic regression, doesn't provide classes probabilities.

Training an SVM classifier requires parameters tuning. The first parameter λ decides how flexible (or how biased) we want the classifier to be. Having a very large λ value yields a high bias/low variance classifier. A low value of λ yields the opposite. This value is related to as how many misclassifications (or how wide the margin) are we allowing: a wide margin (large λ) means many vectors can be on the wrong side of the boundary decision. The second parameter comes from the kernel definition. A discussion regarding the choice of kernel goes beyond the scope of this paper but in our selected radial basis function, we have to pick the variance σ^2. There again, a very wide Gaussian function means higher bias and lower variance, while a small value of σ^2 means the opposite.

In practice, we select the value of σ^2 and λ by estimating many SVM classifiers for a wide range of values of σ^2 and λ (e.g., $[10^{-5}, 10^5]$) and select the values that yield the lowest classification error. Statistically, we perform cross-validation using a dataset called the validation set, different from the training or test sets (see details below).

The SVM classifier is quite similar to logistic regression, as exemplified in the similarities between the two loss functions of Figures 1(b) and 1(a). SVM has a computational edge over logistic regression because only observations within the margins or at the margins matter. Note that kernels could also be easily incorporated into the definitions of logistic regression but without the same computational benefits.

It is interesting to see that a trend using neural networks (NNs) is re-emerging for the interpretation of seismic data (Araya-Polo et al., 2017; Huang et al., 2017). The fact that NNs can be efficiently trained and optimized on hardware accelerators such as GPUs seems to be the most compelling reason why they are being used. However, NNs are also quite expensive to train. We think that determining the right feature sets for the training of the classifier is one of the most important element of MLAs and will benefit any classifier. Furthermore, the right set of features might save us from using very expensive classifiers such as NNs while still obtaining satisfying prediction results. Therefore, we think that SVMs with the appropriate kernels and parameterization combined with meaningful features should yield accurate-enough fault images. Testing other classifiers such as NNs will come at a later time.

Having introduced the classifier, we now describe the feature sets used for the fault imaging problem.

3 FEATURES COMPUTATION FOR FAULT DETECTION

The classifier is only one component of the whole MLA: it helps us deciding whether a training point belongs to a class or another. The next vital component of the MLA is the feature set: what feature in the seismic image are we going to use to make the classification easier?

The most simple feature is the seismic data itself, using amplitude information only. Unfortunately, amplitude is a very poor predictor of faults in seismic data and we need to find better attributes to characterize them. Measures of continuity in seismic data make better predictors, such as semblance. However here, we opted for two classes of features that are widely used in computer vision for object recognition. One is called Scale Invariant Feature Transform (SIFT) (Lowe, 1999), and the other one is called Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005). These two methods yield sets of features that can be used for classification. One of their biggest shortcoming is that they work in 2D slices only: we know that adding more dimensions to the fault detection problem yields better results. One could expend these methods to extract 3D attributes, however.

We are now going to briefly review the SIFT and HOG algorithms.

3.1 Histogram of Oriented Gradients

The computation of HOG features is quite simple and computationally efficient (Dalal and Triggs, 2005). First, an image is decomposed into cells and blocks. Cells are usually half the size (in number of pixels) of blocks. Blocks are mostly used for normalization purposes of the histograms. In general, but not always, cells are 8x8 and blocks 16x16.

The first step consists in computing vertical ∇_z and horizontal ∇_x gradients using centered 1D derivative operators $[-1, 0, 1]$ at each location in the image. From these gradients, signed (between $0^o - 360^o$) and unsigned (between $0^o - 180^o$) angles are computed (e.g. $\alpha = \text{atan}(\nabla_z / \nabla_x)$).

Next, histograms of angles are computed for each cell. In practice, nine orientations bins are usually estimated, where each angle is linearly interpolated between neighboring bins. The value of the bin depends on the gradient magnitude $(\nabla_x^2 + \nabla_z^2)^{0.5}$

Note that it would be possible to extend these computations to 3D by estimating histograms of oriented dips and azimuths (Marfurt, 2006). For dips, we could compute

$$\alpha = \text{atan}\left(\frac{(\nabla_x^2 + \nabla_y^2)^{1/2}}{\nabla_z}\right), \tag{15}$$

and for azimuth

$$\psi = \text{atan2}(\nabla_x, \nabla_y). \tag{16}$$

Incorporating 3D attributes like these will help the classification and should be explored further.

Once histograms have been estimated, a normalization step follows. The normalization compensates for local variations in amplitudes due to illumination effects, noise, geology etc... The normalization is done for each block (i.e., groups of cells) where blocks are overlapped. For each block, a normalization factor is computed as follows: if $\mathbf{v}$ is the unnormalized vector of all histogram values for all cells belonging to a block, compute the normalization factor $\sqrt{\|\mathbf{v}\|_2^2 + \epsilon}$, where ϵ is a constant to be chosen (with little impact on the classification results). Each histogram within a block is then normalized by this value.

This completes the computation of HOG features. Again,

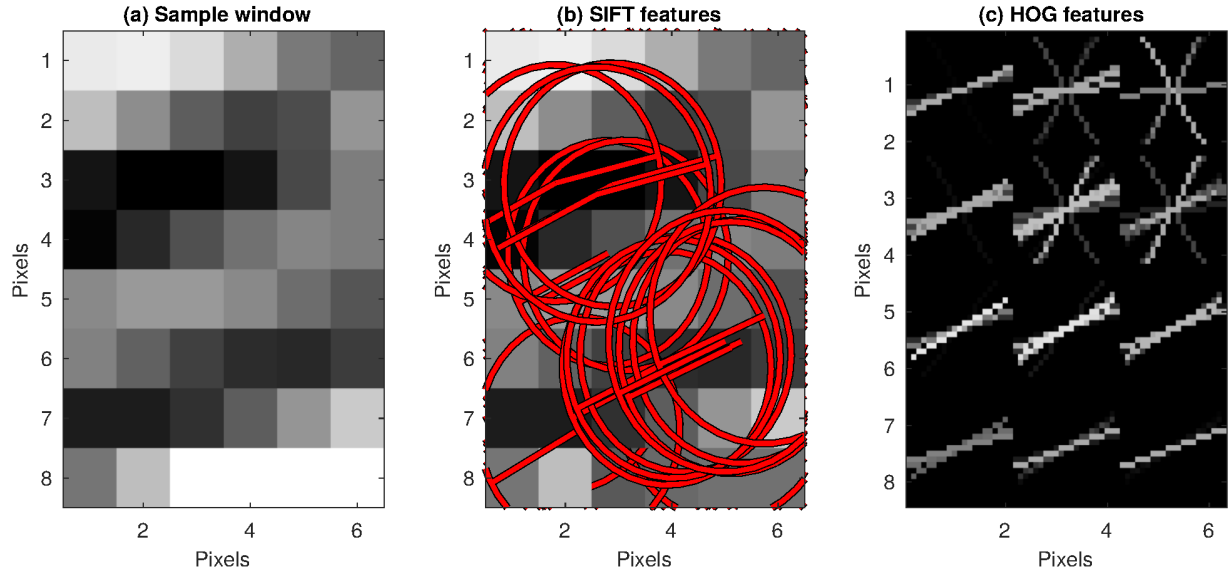


Figure 4. (a) A window extracted from a 2D slice after patching. (b) Illustration of SIFT features extracted from (a): 15 keypoints were identified with different orientations. (c) Illustration of the HOG features extracted from (a): histograms are represented as rose diagrams.

the 3D extension should be easily doable and investigated further. We now look at the Scale Invariant Feature Transform (SIFT).

3.2 Scale Invariant Feature Transform

SIFT is another popular descriptor for image classification. The goal of SIFT is to match features across different images. The power of SIFT is that it can match features between images that are rotated, scaled (size-wise), illuminated or viewed from different angles. The computation of SIFT descriptors is more involved than for the HOG ones. We will focus on the main steps only.

The first step of SIFT consists in finding keypoints in the image. These keypoints are supposed to represent prominent features in the image. Finding these keypoints requires many steps. First, the image is transformed into a scale space: the image is simultaneously blurred with different Gaussian kernels and sub-sampled (halved between scales). Each scale is called an "octave". Therefore, after this first stage, SIFT creates many duplicates of the image at different scales (i.e., resolution) and with different smoothing functions. In other words from one image, we end up with many cubes of different sizes where each panel of a cube has a different blurring kernel applied to it. Then, for each octave, two consecutive blurred images are subtracted to create a Difference of Gaussians (DoG) scale space. In the next step, local extrema (minima and maxima) are detected by looping over all pixels for each scale within an octave and comparing it with its closest 26 neighbors (a pixel has 26 neighbors in a cube). The local extrema are then quadratically interpolated so that we have a value of extrema everywhere. It is recommended to have two such images of extrema per octave, which means four DoG images per octave and five blurring kernels. Extrema with low contrast responses and close to the edges are discarded.

Once the keypoints are identified at different scales, an orientation for each keypoint is estimated. To do this, a local dip is estimated for all points surrounding a keypoint using the same formula as the one used for the HOG method. Then a histogram of all dip values is computed. The bin with the highest value is assigned as the keypoint dip. Other keypoints with the same location and scale are created for all bins within 80% of the highest value.

Finally, for each keypoint 16 blocks (4x4) are defined with 16 pixels per block (4x4). Within each block, histograms of gradient magnitudes and orientations are computed and binned in an 8 bins histogram (similar to HOG). The keypoint orientation is subtracted from all computed orientations within surrounding blocks to preserve rotation independence. The bin values are also clipped for normalization. Therefore, for each keypoint, we have a vector of 128 descriptors (16 blocks times 8 bins). For classification, the descriptor of 128 elements is used rather than its location.

Some of the major differences between the HOG and SIFT descriptors are the scale invariance property built in the SIFT, not present in HOG, and the selection of keypoints. For each image, the number of keypoints will be different and more steps are needed to use the SIFT descriptor with MLA. We detail these steps below.

3.3 Clustering of SIFT features

From the description of the SIFT descriptor above, it is clear that different images will have a different number of keypoints.

This is problematic with MLAs because we want the size of the feature vector to be the same for each data point.

To remedy the fact that the number of keypoints will be different for different images, we first estimate the SIFT features for each training image. For each image i, we end up with a descriptor $D_{128\times M_i}$ where M_i is the number of keypoints for image i and 128 is the number of orientation bins (8) times the number of blocks (16) surrounding the keypoint. Then, we pack all the descriptors for all images into one large matrix $\mathbf{D}$ of size $128 \times \sum_i M_i$.

The matrix $\mathbf{D}$ contains all vectors of size 128 for all training images. In the next step, we classify the vectors using K-means clustering, an unsupervised learning algorithm. The number of centroids doesn't need to be large. For the fault detection problem, 20 centroids seem to be enough. Once the centroids in the 128 dimensional space are found, we classify all descriptors for all images into the 20 classes (using smallest distance). If, for one image, l descriptors fall into one class, the class takes the value l. We end up for each image with a histogram of centroid values that has the same size for all images. If no keypoint is found in one image, then the histogram is zero for all 20 bins. Another advantage of this process is that it reduces the size of the feature vector considerably (only 20 elements in our case).

The workflow is now almost complete. We compute HOG and SIFT features from seismic images. SIFT features are simplified by clustering. The features are fed into an SVM classifier for training. The SVM classifier is then used for classification. Now, we present our workflow for the fault detection problem first with 3D synthetic data, then with 3D field data.

4 3D STATISTICAL FAULT DETECTION WITH SYNTHETIC DATA

The proper training, parameterization and execution of an MLA requires validation, training, and test sets. These three sets are mutually exclusives and don't contain the same data points. For seismic interpretation, we have enough data to properly build these easily. The MLA is trained on the training dataset and its parameterization is tested on the validation set. For the SVM classifier with a Gaussian kernel, two parameters are estimated with 5-fold cross-validation: σ (equation 13) and λ (equation 9). Once the parameters are estimated, the training is performed using the training set and the classifier is applied to the test set.

4.1 Building the outcome vector

For the fault imaging problem, we first need to build the required sets of labeled seismic data. For this, we build many synthetic datasets with different fault geometries (normal, conical, conjugate) and pick them using the Mines Java Toolkit (https://github.com/dhale/jtk) and the seismic image processing for geological faults software (https://github.com/dhale/ipf). The fault picking part is explained in details in Hale (2013). For the training and validation sets, we use the volumes in Figures 2(a), 2(b) and 2(c). We build the features vectors $\mathbf{x}_j$ from the seismic volumes and the outcome vector $\mathbf{y}$ from the fault picks. We now explain how these features are extracted.

4.2 Building the feature vectors

The HOG and SIFT features need to be estimated on small windows. Therefore, we decompose each cubes into 2D slices along the crossline direction. Each slice is then decomposed into overlapping windows of 8x6 pixels, with an overlap zone of half the window length in both directions. We estimate the HOG and SIFT features from these small windows. In this parameterization, each patch (or window) becomes one training/validation point.

For the outcome vector, we follow the same idea. Figure 3 shows how the patching is done when the window size is 2x2 (for illustration purposes only). In Figure 3a, we have a masking function equal to 1 or 0 when a fault is present or not. In Figure 3b, patches of size 2x2 are formed. If a fault is contained in a patch, we assign the value 1 to the whole window, 0 otherwise (Figure 3c): this last step gives us the outcome vector for each patch. Going backward, reassembling the patches back to the original grid, we end up with Figure 3d: due to the patch size and overlap, the fault location is smoothed across the 2D slice as shown by the numbers in the cells (in practice, the cells are then re-normalized so that the maximum value is one).

From the patching of the 2D seismic slices, we extract the individual windows and estimate the HOG and SIFT features. Figure 4 illustrates these features. Figure 4a shows a patch of size 8x6. From this patch SIFT features are extracted and displayed in Figure 4b. For this window, 15 keypoints are identified, corresponding to 15 circles. The size of the circle corresponds to the scale, or octave, the feature was identified in. The bars correspond to the orientations picked at the keypoint locations. Other windows will have different number of keypoints, orientations and circle sizes (i.e., scale). The clustering algorithm presented above remedies this issue and provides us with 20 features for all points. The HOG features are displayed in figure 4c. The histograms of oriented gradients are represented as rose diagrams. When slopes are locally consistent, the histogram is very narrow and highlights a fairly uniform direction. For each window (patch), 12 histograms are estimated for a total of 768 features.

From 3D volumes of interpreted faults and seismic data, we build 2D patches. Each patch makes up for one data point. From the fault mask patches, we build the outcome vector $\mathbf{y}$. From the seismic data patches, we build the feature vectors $\mathbf{x}_j$ using HOG and SIFT algorithms. The final size of each feature vector is 788 (768 from HOG, 20 from the clustering of the SIFT vectors). With all our labeled data and features sets available, the training of the SVM classifier can start. In the next sections, we show our training/prediction results with different combinations of features (HOG vs. SIFT vs. HOG+SIFT).

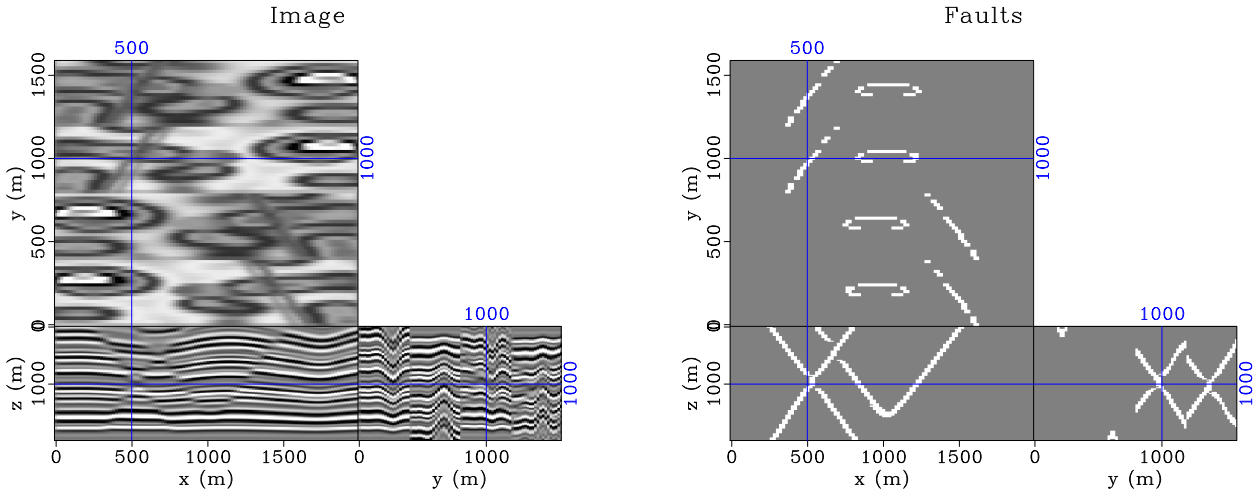


Figure 5. Test data with its interpreted faults. This dataset comprises conical, conjugate and normal faults. These data are not used for training or cross-validation.

Figure 6. Confusion matrices for (a) the training data and (b) the test data when the HOG features only are used. The error rate for the training data is very low at 0.6% (bottom right blue corner). The error rate for the test data is higher, as expected, at 7.9%. Looking more closely at the second row of (b), we notice that our classifier tends to overpredict faults, with an error rate of 20.8% (meaning that 20.8% of predicted faults are not faults).

4.3 Fault interpretation with HOG features only

We first train and predict using the HOG features only. The training is done using the seismic volumes in Figures 2(a), 2(b) and 2(c). After patching, 600,000 windows are available for validation and training. We extract randomly 40,000 patches for training and 7,500 for validation (i.e. parameterization of the SVM classifier with Gaussian kernels).

The training outcome is displayed in the confusion matrix of Figure 6(a). The confusion matrix summarizes the training error by displaying the true/false positive/negative ratios. The diagonal elements show the true positive and true negative rates. The off-diagonal elements show the false positive and false negative rates (classification errors). The bottom row and right-most column show a summary (in percentage) of the classification errors/successes. The bottom right corner (blue) shows in green the total success rate (sum of diagonal elements) and in red the total misclassification rate (sum of off-diagonal elements). The training here is very good, with a very low classification error. Only 245 patches with faults were misclassified as not having faults, while only 2 patches without faults were misclassified as having faults. Such a low error

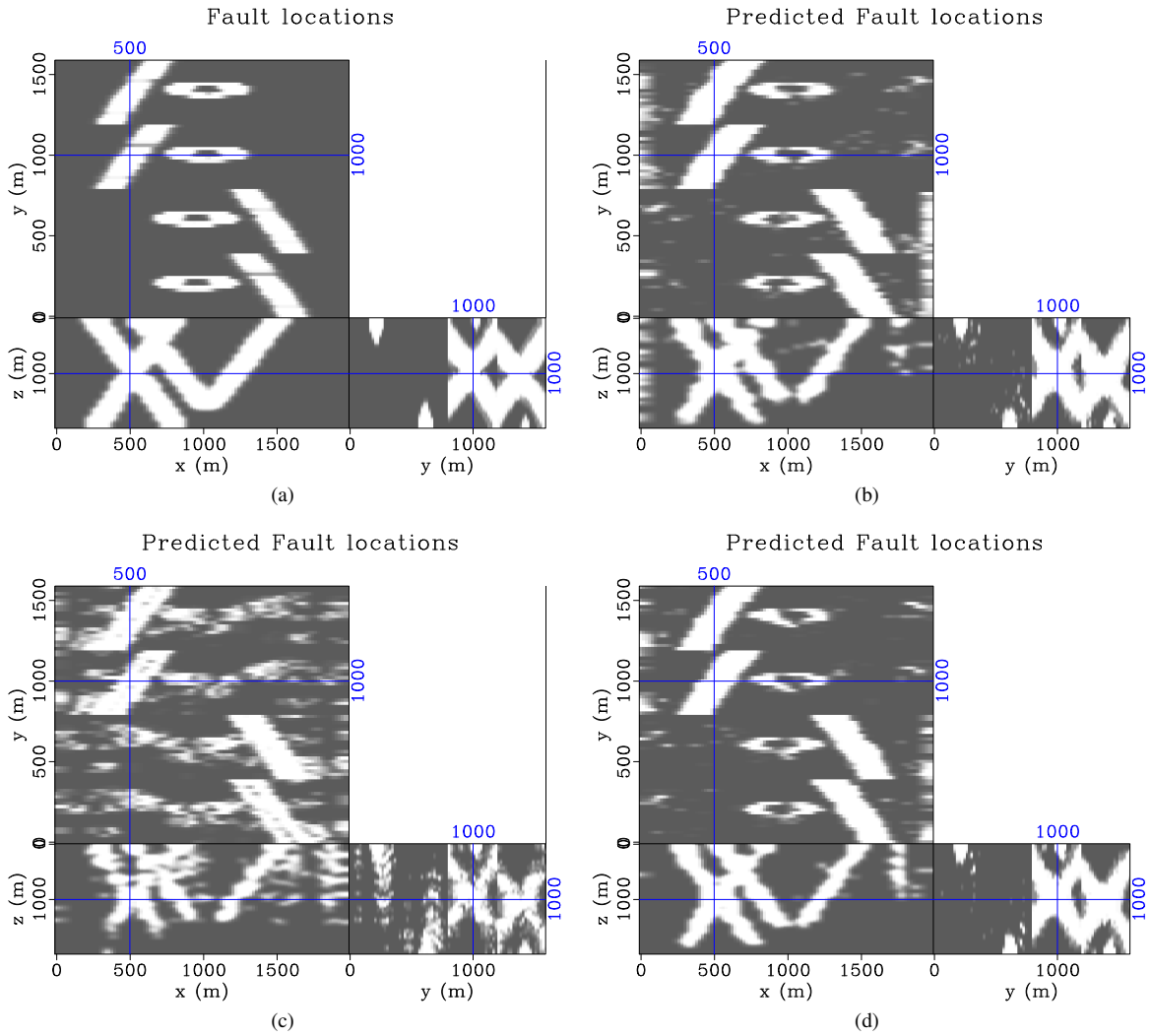
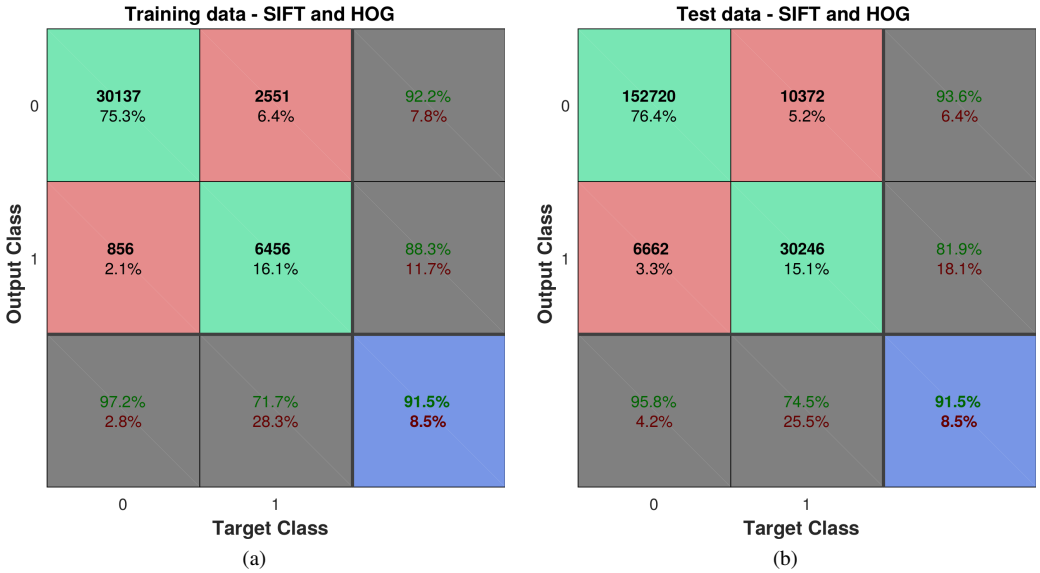


Figure 7. (a) True fault locations after reassembling the patches (and thus smoother than in Figure 5) and (b) predicted faults locations for the test data when HOG features are used only. The prediction highlights the faults accurately with some edge effects. (c) Predicted fault locations for the test data when SIFT features are used only. The prediction is quite noisy and not as accurate as in Figure 7(b). (d) Predicted fault locations for the test data when HOG and SIFT features are used. The prediction is cleaner than in Figure 7(b) with fewer misinterpreted faults on the edges of the cube.

rate might indicate an overfitting of the training data which we could remedy by adding more training data.

After training, we use the SVM classifier to identify faults in a seismic volume not seen during training or validation (Figure 5). We use 200,000 windows for the test data. Figure 6(b) shows the confusion matrix for the test data. The classification error increases significantly, as expected. Looking at the second row of the confusion matrix, we notice that 20% of the interpreted faults are misclassified. Looking at the first row, only 4.5% of the non-faults are misclassified. Therefore, our classifier tends to over-predict faults.

Figure 7(a) shows the interpreted faults after reassembling the patches of the faults images in Figure 5. The patching makes the faults wider, as already explained in Figure 3. We consider Figure 7(a) to be the answer, or true prediction, of the fault locations. Figure 7(b) displays the locations of our predicted faults using the HOG features only. There is a very good agreement between Figures 7(a) and 7(b). We notice some edge effects that would be mitigated with a proper taper function. Overall, the classifier was able to identify all faults properly. Given that the feature vector is computed on 2D windows only, we think that better results would be possible by estimating features in 3D. Yet, the classifier performed remarkably well.



(a) (b)

Figure 8. Confusion matrices for (a) the training data and (b) the test data when the HOG and SIFT features are used. The error rate for the training data has increased to 8.5% (bottom rights blue corner). Looking more closely at the second row of (b), we notice that our classifier tends to predict fewer faults than in Figure 6(b) with a lower error rate of 18.1%. Predicting fewer faults translates into a cleaner image.

4.4 Fault interpretation with SIFT features only

Now we train our SVM classifier using the SIFT features only. For this case, we show the classification results only in Figure 7(c). Clearly, the classification error is high. Although not shown here, the confusion matrix indicates that 37% of faults in Figure 7(c) are misclassified. Increasing the number of centroids in the clustering part of the SIFT vectors didn't change this rate.

4.5 Fault interpretation with HOG and SIFT features

We are now combining HOG and SIFT features to train our classifier. The classification result is shown in Figure 7(d) and seems to indicate a better classification than when HOG features are used only. Looking at the confusion matrices for the training and test data in Figures 8(a) and 8(b), we notice that the overall training error has increased to 8.5% compared to the 0.6% seen Figure 6(a). However, the test error has remained pretty constant. Using HOG and SIFT features has reduced the variance of our classifier, decreased the misclassifications of faults by 2%, and increased the misclassifications of no-faults by 2% as well (precision has increased, recall has decreased). In other words, our prediction has become more conservative and has predicted fewer faults (42326 with HOG only, 36908 with HOG and SIFT), thus resulting in a cleaner image. Because of the overlap of the patches and the inherent robustness coming with it (the information for one fault is spread among many patches), missing more faults is beneficial to the overall prediction.

Therefore, although counter-intuitive given the numbers in the confusion matrices, combining HOG and SIFT features for the fault imaging problem yields the best results. Better classification results could be obtained if more training data were added: we are only using 40,000 points out of 600,000 to limit the computing cost of the training part.

In the next section, we use our approach to image faults for a 3D field data example where many faults are present.

5 3D STATISTICAL FAULT DETECTION WITH FIELD DATA

Field data are more challenging than synthetic data. Building synthetic training data and applying them to field data might not be the best approach because we might not capture the complexity of faults in 3D with simple models. It seems a better approach to train the classifier with field data examples as well. Ideally, we would want to access lots of labeled seismic cubes with different fault geometries, different noise levels, different geological settings, etc... The classifier, similar to a human interpreter, would learn from all these scenarios and would improve its predictions.

For the field data example therein, we select a small 3D seismic volume instead that we divide in three parts for validation, training and testing. We follow the same procedures as with the synthetic data with two important differences. First, the window size for each patch is now 8x12, as opposed to 8x6 for the synthetic dataset. Finally, a patch has a fault if at least 16 cells in the patch (out of a possible 96) have the fault marker set to one. By limiting the number of patches getting the value $y = 1$, we make our classifier more robust to noise. For the validation step, we use 30,000 points. For the training step, we use 128,000 points. For the testing, we use 147,000 points.

Figure 11(a) shows the test data, not seen by the classi-

fier, and Figure 11(b) displays the labeled faults using Hale's software (and after reassembling the outcome vector patches). Using field data presents an interesting challenge, beyond the noise problem mentioned above. The labeling of the faults is indeed not accurate everywhere. Looking at Figure 11(b) we see fault locations in the test data not picked by the automated fault detection software. It might also happens that picked faults are not true faults. Therefore, the training of the classifier is done with mislabeled data, where false positives and false negatives are present.

We train the SVM with the HOG features only. The confusion matrix for the training part is shown in Figure 9(a). The overall training performance is excellent, with a near-perfect prediction. This indicates an overfit of the training data which often results in a poor performance of the classifier on test data. This can be seen Figure 9(b) where the classification of the test data is indeed less accurate (77%). The outcome vector for the test data mapped back into the original dataset space is shown in Figure 11(c). To increase the vertical and horizontal continuity of faults in the crossline direction, we smooth the fault map of Figure 11(c) and obtain Figure 11(e). It is pleasing to see that the five major vertical faults between x=609 km and x=613 km are identified by the classifier. Their lateral extent in the crossline direction is also clearly predicted by the MLA.

Now, we train the SVM with both HOG and SIFT features. Figure 10(a) shows the confusion matrix for the training part. Adding the SIFT features clearly affects the performances of the training part, with a 86.5% success rate. This is a well-known effect of adding more features to the training of any classifier. Similar to the synthetic experiment, Figure 10(b) shows that adding the SIFT features increases the precision by 4%, which is significant: we are predicting less faults but more accurately. Next, we map the predicted outcomes back into the seismic volume in Figure 11(d). We notice that the main faults are predicted correctly. We can also see that we are missing some faults, comparing with Figure 11(b), but that we are also picking real faults not visible in Figure 11(b). Therefore, our MLA does better in some areas, worse in others. Finally, to improve the continuity of faults in the vertical and crossline directions, we apply a small smoothing to the predicted faults (Figure 11(f)). The smoothing makes for a more realistic fault image. This process should be incorporated somehow inside our training part.

6 DISCUSSION

Overall, our approach to predict fault locations using MLA works. We label faults using an automated procedure developed by Hale. We build feature vectors using standard object recognition methods such as HOG and SIFT. We use an SVM classifier with Gaussian kernels. There are many avenues for improvement, however.

From a MLA view point, we could improve our prediction by incorporating some smoothness in the predictor directly and not in a post-processing step. For this, the method of Wang et al. (2014) using spatially-temporally consistent tensors could be used. In the geophysical world, this comes down to adding a smoothness term to the fitting goal. Another improvement could come by integrating all our features (SIFT and HOG) in a better way by adding a joint-structured sparsity regularization term (Wang et al., 2013). In essence, we can try to identify automatically the best features for all vectors. Finally, we could use a semi-supervised approach where we don't label all faults but just a few: this would save time and might handle the mislabeling issue with field data better.

From a feature computation view point, extending the SIFT and HOG methods to 3D is the next natural step. While the labelling is done in 3D, the features are estimated in 2D planes only. By taking the SIFT and HOG features to higher dimensions, better predictions would follow. In addition, we could use other features such as semblance, dip, etc... for the classification. However, it is quite remarkable that standard object recognition techniques work so well with seismic data.

Additionaly, we need to include more training data to have a classifier able to identify faults in many environments. Increasing the size of the training data requires larger computing cappabilities and more efforts have to be made to optimize the computation of both features and predictor. To this end, being able to run these algorithms efficiently on GPUs will be required.

Finally, the fault imaging problem is only one possible application of MLAs with seismic data. Other features such as channels or sequence boundaries can be included as well. MLAs can also be very useful in the integration of many data types where the volume and complex interaction between them makes it hard to do with standard inversion approaches.

7 CONCLUSION

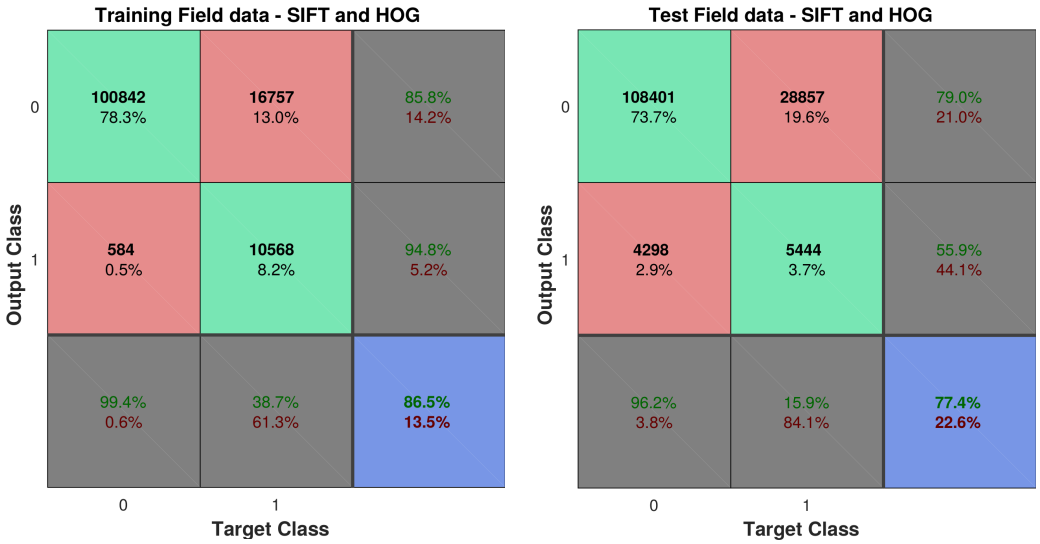
We highlight faults in 3D seismic volumes using a supervised machine-learning approach. We label faults using an automated fault-picking method developed by Hale. We build feature vectors using two methods widely used in object recognition techniques called HOG and SIFT. We use a standard SVM classifier with Gaussian kernels for our predictor. On both 3D synthetic and field data examples, we show that a combination of HOG and SIFT features yields better classification results: the precision increases resulting in a lower false-positive rate. We also demonstrate that a non-accurate labelling of the training data doesn't necessarily prevent the MLA from predicting faults: this is an important message with field data where mislabelling will occur. Finally, we prove that although not optimal, features extracted in 2D can still be useful in 3D contexts. However, we advocate to extend HOG and SIFT to 3D thus yielding better classification results.

REFERENCES

Araya-Polo, M., T. Dahlke, C. Frogner, C. Zhang, T. Poggio, and D. Hohl, 2017, Automated fault detection without seismic processing: The Leading Edge, **36**, 208–214.

(a) (b)

Figure 9. (a) Confusion matrix for the field training data using the HOG features only. The near-perfect training result indicates an overfit of the training data. This often results in poor performance of the classifier on test data. (b) Confusion matrix for the field test data. More realistic numbers are obtained: 77% of data are classified correctly. The precision is around 52%.



(a) (b)

Figure 10. (a) Confusion matrix for the field training data using the HOG and SIFT features. 61% of "true" faults (target) are not labeled correctly. (b) Confusion matrix for the field test data. Similar to the synthetic case, adding more features increases the precision. The classifier has less variance.

Dalal, N., and B. Triggs, 2005, Histograms of oriented gradients for human detection: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01, IEEE Computer Society, 886–893.

Hale, D., 2013, Methods to compute fault images, extract fault surfaces, and estimate fault throws from 3d seismic images: Geophysics, **78**, O33–O43.

Hastie, T., R. Tibshirani, and J. Friedman, 2001, The Elements of Stastical Learning, Data Mining, Inference, and Prediction: Springer.

Huang, L., X. Dong, and T. E. Clee, 2017, A scalable deep learning platform for identifying geologic features from seismic attributes: The Leading Edge, **36**, 249–256.

James, G., D. Witten, T. Hastie, and R. Tibshirani, 2013, An Introduction to Statistical Learning, with Applications in R:

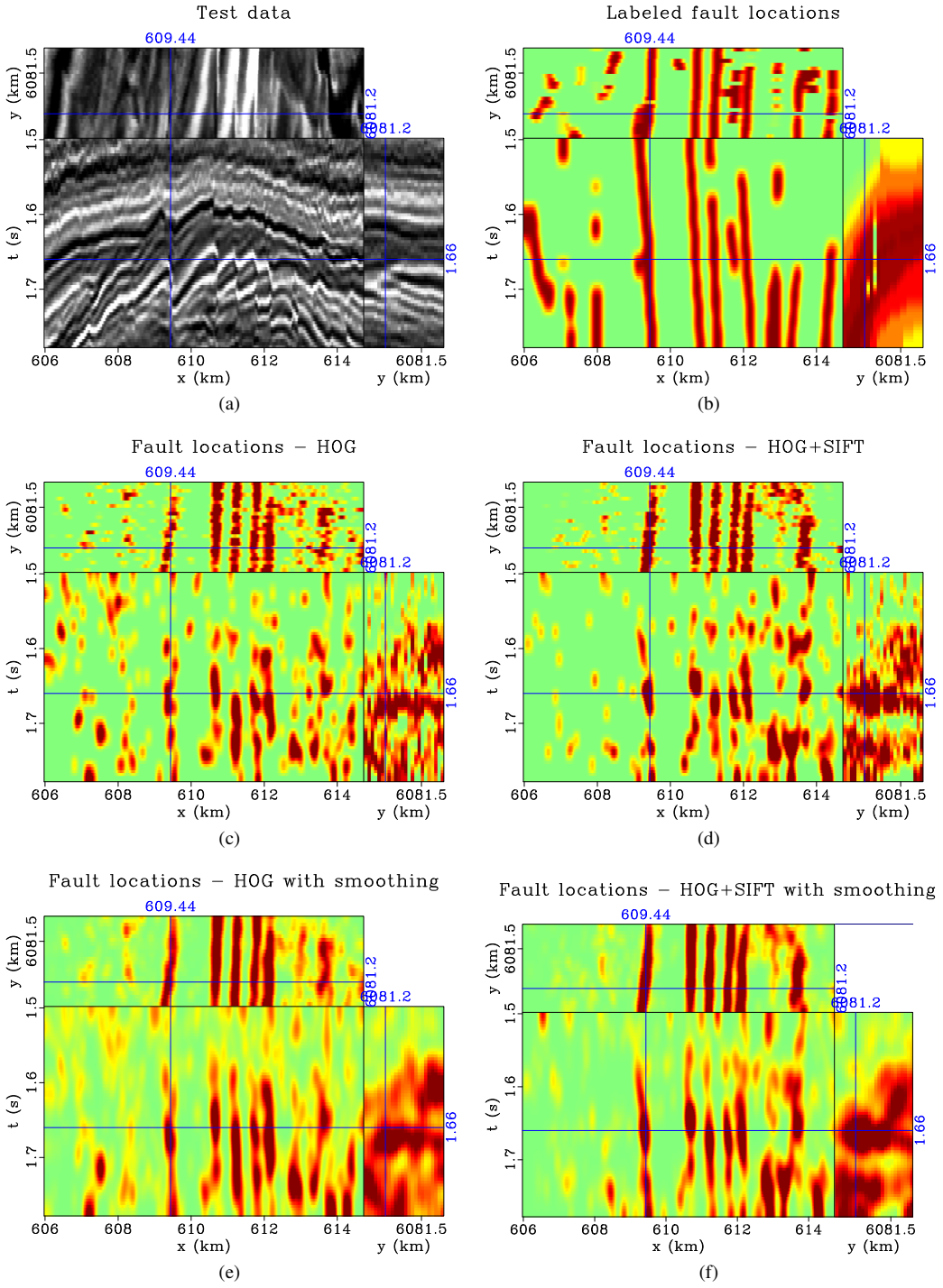


Figure 11. (a) Test data. (b) Fault images picked automatically. Predicted faults using (c) HOG features and (d) HOG+SIFT features. Predicted faults after smoothing for (e) HOG features and (f) HOG+SIFT features. The color scale goes from 0 (green) to 1 (red) and could be interpreted as a fault probability map. Smoothing improves temporal and spatial continuity.

Springer.

Lomask, J., A. Guitton, S. Fomel, J. Claerbout, and A. A. Valenciano, 2006, Flattening without picking: Geophysics, **71**, P13–P20.

Lowe, D. G., 1999, Object recognition from local scale-invariant features: Proceedings of the Seventh IEEE International Conference on Computer Vision, 1150–1157 vol.2.

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

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

Pedersen, S. I., T. Skov, T. Randen, and L. Sønneland, 2005, Automatic fault extraction using artificial ants: *in* Mathematical Methods and Modelling in Hydrocarbon Exploration and Production, Springer Berlin Heidelberg, 107–116.

Stark, T. J., 2004, Relative geologic time (age) volumes-relating every seismic sample to a geologically reasonable horizon: The Leading Edge, **23**, 928–932.

Wang, H., F. Nie, and H. Huang, 2014, Low-rank tensor completion with spatio-temporal consistency: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI Press, 2846–2852.

Wang, H., F. Nie, H. Huang, and C. Ding, 2013, Heterogeneous visual features fusion via sparse multimodal machine: 2013 IEEE Conference on Computer Vision and Pattern Recognition, 3097–3102.