Automatic recognition of seafloor features in sub-bottom profiles using eigenimages

Charles Garcia

Advised by Miles Logsdon

University of Washington, School of Oceanography

1503 NE Boat ST, Seattle, Washington 98105

cggarcia@uw.edu

06/2015

Acknowledgements

Thank you Miles Logsdon for your unwavering support, confidence, and advice over the past two years. Thank you Arthur Nowell for the organization, the motivation, the kind words, and the cupcakes. Thank you to all aboard R/V *Thomas G. Thompson* and ashore in December, 2014 who helped make TN316 a safe, productive, and fun expedition. Finally, thank you to the behind-the-scenes magicians at the School of Oceanography and University of Washington who make these expeditions possible.

Abstract

Sonar is the primary tool used to remotely survey the seafloor. Sub-bottom profilers use pulses of sound to penetrate the seafloor and create an acoustic profile of the seafloor and sub-bottom. The profiles are traditionally inspected by sight to identify features of interest which is time-consuming and requires an experienced human's eye. I have developed a method using eigenimage analysis that can automatically distinguish between three dfferent types of seafloor features found in the fjords of Nootka Sound: Sediment, sills, and rockslides. The method uses a training set of features to build eigenimages that represent the orthogonal variance between different features. Test features are projected onto the eigenimages and compared to the average three feature types. The resulting projection coefficients are a signature characteristic to each feature type and can be used to identify and classify seafloor features.

1. Introduction

Water strongly absorbs most of the electromagnetic spectrum which renders a significant portion of humanity's terrestrial remote sensing toolkit useless beneath the sea. Fortunately sound waves propagate efficiently in water and numerous types of sonar have been developed that help us understand and explore the ocean. Most modern academic research vessels are equipped with several permanent acoustic instruments like multibeam echosounders and sub-bottom profilers that collect more data than could reasonably be processed and analyzed by individuals. It is likely that the gap between collected data and human analysists will continue to grow in the future, especially if remotely operated vehicles (ROVs) and autonomous underwater vehicles (AUVs) begin to carry out more acoustic surveys. There is one practical way to bridge the analysis gap: Automation.

The full automation of processing and analyzing data for even one type of instrument would be a huge undertaking but can be whittled down to manageable portions. I focus on the problem of recognizing broad categories of seafloor features in profiles gathered by a sub-bottom chirp echosounder. The sub-bottom feature recognition problem falls somewhere in between the field of ocean acoustics and computer vision. The more general image recognition problem has been automated repeatedly in many different cases with almost as many different methods. Facial recognition was one of the first cases solved and is a major inspiration behind this thesis. Broadly speaking, faces are not so different from seafloor features. Most faces share a similar structure – a mouth, a nose, eyes, etc. – with minor variations in spacing, sizes, and so on between individual faces. Likewise seafloor features such as seamounts or submarine landslides

have similar fundamental properties – acoustic impedance, roughness, shape – that vary slightly between the same categories of features. It is not too farfetched to suggest that the eigenimage approach first used to automatically distinguish between different faces may be able to distinguish between different seafloor features.

1.1 Eigenimage approach

Eigenimage analysis has a formulation and intuition common to principal component analysis (PCA), empirical orthogonal function (EOF) analysis, and canonical correlation analysis (CCA). The heart of these techniques is the Karhunen-Loève expansion which represents some set of high dimensional objects, maybe one hundred 10 megapixel images or a time series of 10^6 global temperature measurements, as linear combinations of orthogonal basis functions. While the basis functions have the same dimensionality as the objects they represent, the linear combination coefficients typically have a much lower dimensionality equal to the number of basis functions. In the hypothetical example of one hundred 10 megapixel images, each image could be stored as one hundred or fewer expansion coefficients without significant loss of information. The key usefulness of eigenimage analysis is this ability to compress the information contained within high dimensional objects.

The eigenimage approach was first developed to represent and characterize the faces of different individuals (Sirovich & Kirby, 1986). Eigenimages were later combined with face detection and tracking to create the first practical automatic facial recognition software (Turk & Pentland, 1991). Incremental improvements since then have led to methods more sophisticated

than eigenimages (Belhumeur et al. 1997; Yang, 2000; Delac et al. 2007) but inspired by the same basic concept of finding a lower dimensional representation of complex, high dimensional images. The eigenimage approach has also been used to recognize voices (Kuhn et al. 2000), organs in medical tomography (Windham et al. 1988), and other disparate features. In most cases eigenimage analysis seems to be the first practical solution to a specific recognition problem and eventually leads to more sophisticated solutions.

1.2 Sub-bottom profilers

Sub-bottom profilers are echosounders that use a transducer to convert electrical signals to pulses of seafloor penetrating sound. The pulses reflect from objects on or in the seafloor and from the boundaries between layers of different acoustic impedance. The sub-bottom profiler detects the reflected acoustic signals and assigns each returning signal, or trace, an identifier based on the return time and signal characteristics. The echosounder software often does additional signal processing (Gutowski et al. 2002) and may be subject user-adjusted parameters but invariably returns a profile made up of individual signal amplitude and time records, or traces, that approximate the acoustic impedance and thus structure of the seafloor.

Sub-bottom profiles are highly heterogeneous and vary in readability and complexity depending on the instrument and parameters use to collect the data, the methods used to process and display the data, and the physical features of the sampled seafloor. The profiles are typically visually inspected without additional post-processing (e.g. Dove et al. 2014). There are examples example of eigenimage analysis applied to sub-bottom profiles (Kim et al. 2002; Lee et al. 2009)

70 but their approaches characterizes individual traces and does not distinguish between different
71 features made up of multiple traces. Since sub-bottom profiles are already often analyzed by eye
72 as images the eigenimage approach to automatic feature recognition seems appropriate.

2. Methods

The bulk of the study was carried out in four stages. First, sub-bottom profiles were collected, processed, and interpreted. Second, three feature types – sediment, sills, and rockslides – were identified within the sub-bottom profiles and extracted as a training set of 42 images and a testing set of 6 images. Third, 42 eigenimages were computed from the covariance matrix of the training set. Fourth, the 6 testing set images were projected onto the eigenimages and the projection coefficients were compared to the coefficients of three average feature projections.

2.1 Data collection and processing

Sub-bottom profiles were obtained during expedition TN316 on the R/V *Thomas G. Thompson* in December, 2014. Most of the expedition took place in Nootka Sound, a network of fjords on the Southwest coast of Vancouver Island, Canada. The four sub-bottom profiles used in the study were collected during surveys of Muchalat Inlet, Tahsis Inlet, and Tlupana Inlet with a Knudsen 3260 Chirp Echosounder connected to an array of 12 hull-mounted TR-109 3.5 kHz transducers. Multibeam bathymetric surveys, sediment grabs, and shore expeditions were also conducted in the inlets and helped interpret the sub-bottom profiles.

The echosounder was configured to send an outgoing chirp waveform centered at 3.5 kHz with a bandwidth of 3.0 kHz transmitted at a rate of 0.1 s^{-1}, with a pulse length of 6.25×10^{-5} s at a transmission power of 10 kW. Return signals were detected as Hamming filter correlated, square law envelopes in a phase window of 200 m assuming a water sound velocity of 1500 m s^{-1}. The

signals were recorded as traces with 0 dB manual gain, 0 process shift, 0 sensitivity, and time varied gain (TVG) disabled.

Individual traces consisted of 4,444 amplitude values within constant 200 m phase windows. However the beginning of different trace windows was adjusted throughout surveys with the depth of the seafloor. The traces were aligned by padding each individual trace with 0 values to 11,104 total amplitude values per trace corresponding to a window of about 500 m.

2.2 Feature extraction

For the purposes of the eigenimage procedure it is useful to think of the profiles as mathematical objects. In that sense aligned profiles are matrices with approximately 11,104 rows corresponding to the amplitude values within a trace and on the order of 20,000 columns corresponding to individual traces. The three feature types common to each profile – sediment, sills, and rockslides – are extracted by defining 1000 row by 500 column submatrices centered and fit to the top of each feature. The 42 features of the training set to be used for eigenimage computation are extracted from two Muchalat Inlet profiles while the 6 features in the testing set are extracted from profiles of Tahsis Inlet and Tlupana Inlet.

Each extracted 1,000×500 feature matrix is resized to a 100×100 matrix using nearest neighbor interpolation to increase the speed of computations. The feature matrices are then Winsorized by clamping amplitude values at the 95^{th} percentile to lessen the impact of spurious outliers. Finally the matrices are standardized to have a mean of 0 and a standard deviation of 1.

The resized, Winsorized, and standardized feature matrices constitute the 42 images of the training set and the 6 images of the test set.

2.3 Eigenimage analysis and feature recognition

The mathematical formulation of the eigenimage procedure is described fully in appendix 6.1 and the source code is provided in appendix 6.2. The 42 training set images are reshaped from 100×100 matrices to 1×100^2 vectors and concatenated to form a 42×100^2 training set matrix. The $100^2\times100^2$ covariance matrix is then computed from the training set matrix. The 42 eigenvectors and associated eigenvalues of the covariance matrix are computed and ordered by magnitude from largest to smallest. Each $100^2\times1$ eigenvector is then reshaped to a 100×100 eigenimage. Mean images of each feature type in the training set are computed. The mean feature images and the test images are then projected onto the eigenimages resulting in a set of projection coefficients. The L^2 norm between the projection coefficients of each test image and mean image are computed and used as the recognition heuristic.

3. Results

3.1 Sub-bottom surveys

Relative amplitude values are used to interpret the sub-bottom profiles. Low amplitudes are interpreted as soft, low density strata such as mud or diffuse sound scatterers like groups of multiple angular rock fragments. High amplitudes are interpreted as hard, high density consolidated sediment, glacial till, sand, or smooth rock faces. These interpretations are reinforced by co-located multibeam surveys, sediment samples, and shore expedition reports. Most amplitudes fall between of 0 and 3,500. Amplitudes of 0 and 1 correspond to the sound waves' unimpeded transit through the water column. Higher amplitudes signify higher amounts of backscatter from the seafloor and subsurface. Extremely high amplitudes of up to 32,767 are found on the edges of high impedance features and from source-pulse ringing and other water column noise. The high amplitude outliers make up less than 1% of all values and do not appear to be diagnostically useful. Profile plots are capped at amplitude values of 3,500 to increase visual contrast.

The bottom of Muchalat Inlet (Fig. 1) is dominated by layered sediment punctuated by rockslides up to 100 m thick. Two sills obstruct the mouth of Muchalat Inlet, cresting at depths of approximately 100 m. The sediment layers between the sills are about 150 m deep and ten to twenty meters thick. The seafloor descends further up the inlet to 300 m and then again to nearly 400 m. Sediment thickness tends to decrease away from the mouth of the inlet. Tahsis Inlet and Tlupana Inlet (Fig. 2) both have shallow sills at their mouths that then descend to layers of

sediment. However the sills of Tahsis and Tlupana are less symmetric than those of Muchalat. Both inlets also have more gently sloped walls than Muchalat and appear to have fewer rockslides.

3.2 Sub-bottom features

Three types of features consistently appear in the profiles of each inlet: Sediments, sills, and rockslides. Sediment features are identified primarily by at least one layer of low acoustic impedance over a basement of high impedance. Sediment grabs and cores taken from several suspected sediment features confirm that at least the top most layer is correctly identified as soft sediment. Sill features protrude tens of meters from the seafloor, have a defined, often symmetric crest, and a highly reflective perimeter. The locations of the sills identified in the sub-bottom profiles agree with the sill locations charted in each inlet. Rockslides are identified by their characteristic, cumulus cloud-like appearance which is due to the cone of sound emitted from the echosounder scattering from multiple rocks with sharp angles. In many cases the rockslides were also evident above sea level as great gouges in the fjords' steep walls.

The training feature set (Fig. 3) consists of 42 images from two surveys in Muchalat Inlet and contains 31 sediment images, 4 sill images, and 7 rockslide images. The test feature set (Fig. 4) consists of 6 images from a survey in Tahsis Inlet and Tlupana Inlet and contains 2 images of each feature type. The differences between individual sediment images are the number of visible layers, the impedance of the layers, and the shape of the seafloor. About one third of the sediment images are flat, the rest have some significant slope or curvature. Two sills are represented in the four sill images. One sill is slightly narrower than the other and one of the

images is missing a section due to a poorly adjusted phase window. The rockslide images are highly variable. Some capture the crests of rockslides, others are taken in between. The rockslide images are defined by having a fairly even backscatter signal throughout the body of the feature.

3.3 Eigenimages, projections, and norms

The eigenimages and associated eigenvalues (Fig. 5) respectively represent the spatial distribution and magnitude of the covariance between all images in the training set. The first eigenimage accounts for the most covariance in the training set and most closely resembles the flat sediment features. The second, third, and fourth eigenimages resemble sills while the fifth, sixth, and seventh eigenimages resemble a mix of the sediment, sill, and rockslide images. The eighth and later eigenimages appear to converge to roughly resemble sill images with varying pixel patterns. Similarly the first seven eigenvalues decrease rapidly and stagnate after the eighth eigenvalue.

The mean feature images projected onto the eigenimages (Fig. 6) appear to match the original mean images exactly. The test images are not completely recovered from the projections, but do roughly resemble the original features. The two test sediment projections look like flat sediment, but the layers are not clearly visible. Instead there is a smeared band of dark pixels just below the top of each image. The projected test sills and rockslides appear to mainly differ in the distribution of pixels within the perimeter of the features. The sills have a dark outline with light interiors while the rockslides have are similarly shaped but almost uniformly colored. The L2 norms between the test images and the mean feature images are contained in

Table 1. Smaller L2 norms represent a better fit between the test image and the mean feature image.

4. Discussion

4.1 Suitability of the eigenimage approach

The eigenimage approach was used to compute 42 eigenimages and eigenvalues from a training set of 42 images. The first 7 eigenimages appear to account for most of the variance between eigenimages. The first, fifth, sixth, and seventh eigenimages resemble sediment features, the fourth eigenimage resembles a sill, and the second and third eigenimages resemble rockslides. The remaining 35 eigenimages grossly resemble a mix of sill, rockslide, and sediment shapes with minor but complex variations of scattered pixels. I interpret this as meaning that the computed eigenimages can be used to readily distinguish between different major shapes, i.e. sediment vs. sills and rockslides, but that they are not as adept at representing the minor variations between different types of sediment features or between sills and rockslides. I attribute this to the small number of images used in the training set. In facial recognition applications the training sets are typically much larger – on the order of hundreds or thousands of images. I think the eigenimage resolving power would improve within feature types given a large enough training set.

Overall the eigenimage approach seems appropriate for representing and distinguishing between different feature types. The highest weighted eigenimages represent different coherent aspects of features and can be used to readily and almost perfectly reconstruct features within the training set. The linear combinations of eigenimages are not as adept at capturing the intricacies of the test features but the three feature types are still obvious.

4.2 Feature recognition efficacy

The L^2 norm, or Euclidean distance, was used as a metric of whether or not the eigenimages could be used to identify the feature types of test images. Smaller L^2 norms between the projection coefficients a test image and its respective mean feature image's indicate a fit. L^2 norms are thus a naïve method of recognizing the test images as different feature types and the test images should have the lowest L^2 norm with their feature type. This appears to be the case for sediment and rockslide test features. However sill test features had similar L^2 norms to both the mean rockslide image and the mean sill image. I did not expect this outcome and suspect it may be a spurious side effect of using the L^2 norm. The projected mean sill and mean rockslide appear dissimilar and have different projection coefficients. The projected test sills and test rockslides also appear dissimilar and have different projection coefficients. The L^2 norm may fail in this case simply because it is measuring the distance between each projection coefficient and that the distances between all of the test and mean image coefficients just happen to be similar. If so, another metric should be able to correctly distinguish the test sills from rockslides.

4.3 Future improvements

The first major improvement to the facial recognition problem was made by combining eigenimage recognition with face detection and tracking (Turk and Pentland, 1991). Face detection and tracking roughly correspond to automatically detecting and extracting interesting features in sub-bottom profiles, comparing them to an existing set of eigenimages, and then

adding them to that set. Automatic feature detection and extraction could quickly increase the number and diversity of features in a training image set by adding features from the numerous existing sub-bottom datasets. Manual feature extraction is time-consuming by comparison and is the most obvious part of the procedure to improve.

After automating the feature extraction process and increasing the number and diversity of potential training images and eigenimages, the next step would be to test different feature recognition metrics. The L^2 norm is adequate for a first formulation but was quickly done away with in facial recognition in favor of more sophisticated metrics or groups of metrics. Additionally most modern, commercial image recognition systems use eigenimages, if at all, as only one part of a process of several techniques such as Fisherfaces or the Viola-Jones algorithm (Viola & Jones, 2004). The eigenimage approach works for recognizing sub-bottom features but is only the tip of the sill when it comes to computer vision

5. Appendices

5.1 Eigenimage formulation

Each individual image is an $r \times c$ matrix with scalar-valued elements $\theta(x_1, x_2)$ where $1 \leq x_1 \leq r \in \mathbb{Z}$ and $1 \leq x_2 \leq c \in \mathbb{Z}$. The training set of k images is $\mathbf{M}_i$, $1 \leq i \leq k \in \mathbb{Z}$ where

$$\mathbf{M}_i = \begin{bmatrix} \theta_i(1,1) & \theta_i(1,2) & \cdots & \theta_i(1,c) \\ \theta_i(2,1) & \theta_i(2,2) & \cdots & \theta_i(2,c) \\ \vdots & \vdots & \ddots & \vdots \\ \theta_i(r,1) & \theta_i(r,2) & \cdots & \theta_i(r,c) \end{bmatrix}$$

Each $r \times c$ training image $\mathbf{M_i}$ is reshaped to form a $1 \times rc$ vector $\mathbf{M}_i^*$ where

$$\mathbf{M}_i^* = [\theta_i(1,1) \quad \theta_i(1,2) \quad \cdots \quad \theta_i(1,rc)]$$

Each reshaped image $\mathbf{M}_i^*$ is Winsorized so that $\theta_i(x_1, x_2) \leq \theta_{i,\alpha}(x_1, x_2)$ where $\theta_{i,\alpha}(x_1, x_2)$ is the element value at the percentile α of $\mathbf{M}_i^*$.

The Winsorized images $\mathbf{M}_i^*$ are then standardized so that

$$\mathbf{M}_i^\dagger = \frac{\mathbf{M}_i^* - \mu_i}{\sigma_i}$$

where

$$\mu_i = \frac{1}{rc}\sum_{1}^{r}\sum_{1}^{c}\theta_i(x_1, x_2) \text{ and } \sigma_i = \frac{1}{rc}\sum_{1}^{r}\sum_{1}^{c}|\theta_i(x_1, x_2) - \mu_i|$$

The k vectors $\mathbf{M}_i^{\dagger}$ are vertically concatenated to form the $k \times rc$ training image matrix $\mathbf{B}$ where

$$\mathbf{B} = \begin{bmatrix} \mathbf{M}_1^{\dagger} \\ \mathbf{M}_2^{\dagger} \\ \vdots \\ \mathbf{M}_i^{\dagger} \end{bmatrix} = \begin{bmatrix} \theta_1(1,1) & \theta_1(1,2) & \cdots & \theta_1(1,rc) \\ \theta_2(1,1) & \theta_2(1,2) & \cdots & \theta_2(1,rc) \\ \vdots & \vdots & \ddots & \vdots \\ \theta_i(r,1) & \theta_i(1,2) & \cdots & \theta_i(r,rc) \end{bmatrix}$$

Which has a square $rc \times rc$ covariance matrix $\mathbf{C} = \mathbf{B}^{\mathrm{T}}\mathbf{B}$ with k eigenvalues λ_i and k $1 \times rc$ eigenvectors $\mathbf{V}_i$ satisfied by $\mathbf{C}\mathbf{V}_i = \lambda_i \mathbf{V}_i$.

The eigenvectors are then reshaped to k $r \times c$ eigenimages $\boldsymbol{E}_i$.

5.2 Source code

```
%  The following script is meant to be used with four sub-bottom profiles
%  collected during expedition TN316 on the R/V Thomas G. Thompson to
%  Nootka Sound.
%
%  The script is broken up into 7 sections.
%
%   1. Profile alignment
%       Aligns the traces from survey SEG-Y files into coherent profiles.
%
%   2. Training image generation
%       Selects 42 training images from two repeat surveys of Muchalat
%       Inlet.
%
%   3. Mean feature image computation
%       Computes the mean image of the 3 feature types - sediment, sill,
%       and rockslide - represented in the training set.
%
%   4. Test image generation
%       Selects 3 images of each feature type from the surveys of Tahsis
%       and Tlupana.
%
%   5. Eigenimage computation
%       Computes the eigenvectors, eigenvalues, and eigenimages of the
%       training image covariance matrix.
%
%   6. Mean image projection
%       Projects the images of mean feature types onto the training set
%       eigenimages and returns the projections and projection
%       coefficients.
%
%   7. Test image projection
%       Projects the test images onto the training set eigenimages and
%       returns the projections and projection coefficients.
%
%   8. L2/Euclidean norms
%       Computes the L2 norms between each test image and each mean image.
%
% Charles Garcia, 2015, cggarcia@uw.edu

%% 1. Profile alignment
    muchalat1 = alignProfile('muchalat1.sgy');
    muchalat2 = alignProfile('muchalat2.sgy');
    tahsis1   = alignProfile('tahsis.sgy');
    tlupana1  = alignProfile('tlupana.sgy');

%% 2. Training image generation
% Select coordinates of 500x1000 rectangles around features
    [trainingSetCoords,           ...
     MU1Sed, MU1Sill, MU1Slide,   ...
     MU2Sed, MU2Sill, MU2Slide] = ...
     selectTrainingSet;

% Resize 500x1000 features to 100x100 using nearest neighbor interpolation
   [trainingImages,                                ...
```

```
     MU1SedImages, MU1SillImages, MU1SlideImages,  ...
     MU2SedImages, MU2SillImages, MU2SlideImages]  ...
     = resizeImages(muchalat1, muchalat2,          ...
                    MU1Sed, MU1Sill, MU1Slide,     ...
                    MU2Sed, MU2Sill, MU2Slide,     ...
                    100, 100, 'nearest');

% Winsorize images to the 95th percentile value
   trainingImages = winsorize(trainingImages,95);

% Standardize images to have a mean of 0 and standard deviation of 1
   trainingImages = standardize(trainingImages);

%% 3. Mean feature image computation
   meanImages =                                              ...
   computeMeanImages(trainingImages,                         ...
                     MU1SedImages,MU1SillImages,MU1SlideImages, ...
                     MU2SedImages,MU2SillImages,MU2SlideImages);

% Standardize images to have a mean of 0 and standard deviation of 1
   meanImages = standardize(meanImages);

%% 4. Test image generation
% Select coordinates of 500x1000 rectangles around features
    [testSetCoords,               ...
     TA1Sed, TA1Sill, TA1Slide,   ...
     TL1Sed, TL1Sill, TL1Slide] = ...
    selectTestSet;

% Resize 500x1000 features to 100x100 using nearest neighbor interpolation
   [testImages,                                  ...
    TA1SedImages, TA1SillImages, TA1SlideImages, ...
    TL1SedImages, TL1SillImages, TL1SlideImages] ...
    = resizeImages(tahsis1, tlupana1,            ...
                   TA1Sed, TA1Sill, TA1Slide,    ...
                   TL1Sed, TL1Sill, TL1Slide,    ...
                   100, 100, 'nearest');

    testImages(6) = [];
    testImages(3) = [];


% Winsorize images to the 95th percentile value
   testImages = winsorize(testImages,95);

% Standardize images to have a mean of 0 and standard deviation of 1
   testImages = standardize(testImages);

%% 5. Eigenimage computation
    [eigenImages, eigenvalues, eigenvectors] = ...
    computeEigenImages(trainingImages);

%% 6. Mean image projection
    [projMeanImages, projMeanCoefficients] = ...
    projectImages(meanImages, eigenvectors);

%% 7. Test image projection
```

```
114       [projTestImages, projTestCoefficients] = ...
115       projectImages(testImages, eigenvectors);
116
117   %% 8. L2/Euclidean norms
118
119       for n = 1:size(projMeanImages,2)
120           for m = 1:size(projTestImages,2)
121        euclideanNorm(n,m) = norm(projMeanImages{n} -
122   projTestImages{m})/norm(projMeanImages{n});
123           end
124       end
```

```
function profileAligned = alignProfile(filename)

%alignProfile
%
% Usage:
% alignedProfile = alignProfile('filename')
%
% Aligns a profile in SEG-Y file format from a Knudsen Chirp 3260
% sub-bottom profiler.
%
%     ReadSegy() depends on the SegyMAT library by Thomas Mejer Hansen
%     available at http://segymat.sourceforge.net/
%
% Charles Garcia, 2015, cggarcia@uw.edu

[data, segyTraceHeaders] = ReadSegy(filename);

phaseStart = [segyTraceHeaders.TransductionUnit];
phaseWidth = [segyTraceHeaders.TransductionConstantMantissa];

[rows, columns] = size(data);

profileAligned = zeros(2.5*rows-6, columns);

for n = 1:columns;
    profileAligned(:,n) = ...
        vertcat(zeros(((phaseStart(n)/10)*222),1), ...
                data(:,n), ...
                zeros((((500-phaseWidth(n)-phaseStart(n))/10)*222),1));
end
```

```
function imagesWinsorized = winsorize(images,percentile)

%winsorize
%
% Clamps amplitudes of images in a cell array to a percentile of the
% amplitude values in each image.
%
%   Usage example:
%   trainingImages = winsorize(trainingImages,95)
%
%   Clamps the amplitude of each image in trainingImages to the value
%   at the 95th percentile.
%
% Charles Garcia, 2015, cggarcia@uw.edu


nImages = size(images,2);

[rows, columns] = size(images{1});

imagesWinsorized = images;

for n = 1:nImages
    winsorClamp = prctile(reshape(images{n}, 1, rows.*columns), ...
                          percentile);
    imagesWinsorized{n}(imagesWinsorized{n}>winsorClamp) = winsorClamp;
end
```

```
function imagesStandardized = standardize(images)

%standardize
%
% Standardizes each image to a mean of 0 and standard deviation of 1.
% The standardization z of some dataset x is z = (x - mean(x)) ./ std(x)
%
%   Usage example:
%   trainingImages = standardize(trainingImages);
%
%   Each image in trainingImages will be standardized to have a mean of 0
%   and a standard deviation of 1.
%
% Charles Garcia, 2015, cggarcia@uw.edu


nImages = size(images,2);

[rows, columns] = size(images{1});

imagesStandardized = images;

for n = 1:nImages
    imagesMean = mean(reshape(images{n}, 1, rows.*columns));
    imagesStdev = std(reshape(images{n}, 1, rows.*columns));
    imagesStandardized{n} = (images{n} - imagesMean)./imagesStdev;
end
```

```
function [trainingSetCoords,        ...
          p1Sed, p1Sill, p1Slide    ...
          p2Sed, p2Sill, p2Slide] = ...
          selectTrainingSet

%selectTrainingSet
%
% Defines coordinates for ~500x1000 rectangles around features in the
% first and second sub-bottom profiles of Muchalat Inlet (MU1, MU2)
% for the eigenimage training set.
%
%   Usage example:
%   [trainingSetCoords,           ...
%    MU1Sed, MU1Sill, MU1Slide,   ...
%    MU2Sed, MU2Sill, MU2Slide] = ...
%    selectTrainingSet
%
% Charles Garcia, 2015, cggarcia@uw.edu

% Muchalat profile 1 feature coordinates
    % Sediment
        p1Sed(1,1:4)   = [1     500   4150 5150];
        p1Sed(2,1:4)   = [500   1000  4250 5250];
        p1Sed(3,1:4)   = [2400  2900  3450 4450];
        p1Sed(4,1:4)   = [2900  3400  3300 4300];
        p1Sed(5,1:4)   = [3400  3900  3250 4250];
        p1Sed(6,1:4)   = [3900  4400  3350 4350];
        p1Sed(7,1:4)   = [4400  4900  3350 4350];
        p1Sed(8,1:4)   = [4900  5400  3200 4200];
        p1Sed(9,1:4)   = [8450  8950  6350 7350];
        p1Sed(10,1:4)  = [8950  9450  6300 7300];
        p1Sed(11,1:4)  = [9800  10300 6400 7400];
        p1Sed(12,1:4)  = [10300 10800 6400 7400];
        p1Sed(13,1:4)  = [10800 11300 6250 7250];
        p1Sed(14,1:4)  = [11300 11800 6100 7100];
        p1Sed(15,1:4)  = [14000 14500 8100 9100];
        p1Sed(16,1:4)  = [14500 15000 8000 9000];
        p1Sed(17,1:4)  = [15000 15500 8050 9050];
        p1Sed(18,1:4)  = [15500 16000 7900 8900];
        p1Sed(19,1:4)  = [16800 17300 8200 9200];
        p1Sed(20,1:4)  = [17300 17800 8150 9150];
        p1Sed(21,1:4)  = [17800 18300 8000 9000];
    % Sills
        p1Sill(1,1:4)  = [1350  1850  2200 3200];
        p1Sill(2,1:4)  = [5600  6100  1600 2600];

    % Rockslides
        p1Slide(1,1:4) = [7650  8150  1900 2900];
        p1Slide(2,1:4) = [11800 12300 4750 5750];


% Muchalat profile 2 feature coordinates
    % Sediment
        p2Sed(1,1:4)   = [1     500   7150 8150];
        p2Sed(2,1:4)   = [500   1000  7100 8100];
        p2Sed(3,1:4)   = [2100  2600  7700 8700];
```

```
57          p2Sed(4,1:4)   = [3800  4300  8100 9100];
58          p2Sed(5,1:4)   = [11200 11700 6200 7200];
59          p2Sed(6,1:4)   = [14300 14800 3200 4200];
60          p2Sed(7,1:4)   = [14800 15300 3400 4400];
61          p2Sed(8,1:4)   = [15300 15800 3300 4300];
62          p2Sed(9,1:4)   = [15800 16300 3300 4300];
63          p2Sed(10,1:4)  = [17650 18150 4300 5300];
64      % Sills
65          p2Sill(1,1:4)  = [13650 14150 1550 2550];
66          p2Sill(2,1:4)  = [17025 17525 2200 3200];
67      % Rockslides
68          p2Slide(1,1:4) = [3300  3800  3650 4650];
69          p2Slide(2,1:4) = [4900  5400  6500 7500];
70          p2Slide(3,1:4) = [5450  5950  3700 4700];
71          p2Slide(4,1:4) = [6000  6500  5500 6500];
72          p2Slide(5,1:4) = [8500  9000  4000 5000];
73
74  trainingSetCoords = vertcat(p1Sed,   p2Sed, ...
75                              p1Sill,  p2Sill, ...
76                              p1Slide, p2Slide);
77
```

```
function meanImages =                                                    ...
         computeMeanImages(images,                                       ...
                           p1SedImages, p1SillImages, p1SlideImages, ...
                           p2SedImages, p2SillImages, p2SlideImages)

% Compute mean images of each feature type, sediment, sill, and rockslide
% from a set of training images.
%
%   Usage example:
%   meanImages =                                                    ...
%   computeMeanImages(trainingImages,                               ...
%                     MU1SedImages, MU1SillImages, MU1SlideImages, ...
%                     MU2SedImages, MU2SillImages, MU2SlideImages);
%
% Charles Garcia, 2015, cggarcia@uw.edu

nImages = size(images,2);
[rows, columns] = size(images{1});

% Mean sediment
    nSedImages = size(p1SedImages, 2) + size(p2SedImages, 2);
    meanSed = zeros(rows, columns); % preallocate memory

    for n = 1:nSedImages
        meanSed = meanSed + images{n};
    end

    meanSed = meanSed ./ (nSedImages);

% Mean sill
    nSillImages = size(p1SillImages, 2) + size(p2SillImages, 2);
    meanSill = zeros(rows, columns); % preallocate memory

    for n = (nSedImages + 1):(nSedImages + nSillImages)
        meanSill = meanSill + images{n};
    end

    meanSill = meanSill ./ nSillImages;



% Mean rockslide
    nSlideImages = size(p1SlideImages, 2) + size(p2SlideImages, 2);
    meanSlide = zeros(rows, columns); % preallocate

    for n = (nSedImages + nSillImages + 1):(nSedImages + nSillImages + ...
                                            nSlideImages)
        meanSlide = meanSlide + images{n};
    end

    meanSlide = meanSlide ./ nSlideImages;

meanImages{1} = meanSed;
meanImages{2} = meanSill;
meanImages{3} = meanSlide;
```

```
function [testSetCoords,            ...
          p1Sed, p1Sill, p1Slide    ...
          p2Sed, p2Sill, p2Slide] = ...
         selectTestSet

%selectTestSet
%
% Defines coordinates for ~500x1000 rectangles around features in
% sub-bottom profiles of Tahsis Inlet and Tlupana Inlet (TA1, TL1).
%
%   Usage example:
%   [testSetCoords,               ...
%    TA1Sed, TA1Sill, TA1Slide,   ...
%    TL2Sed, TL2Sill, TL2Slide] = ...
%   selectTestSet
%
% Charles Garcia, 2015, cggarcia@uw.edu

% Sediment
        p1Sed(1,1:4)   = [12000   12500   4450 5450];
        p1Sed(2,1:4)   = [14200   14700   4200 5200];
    % Sills
        p1Sill(1,1:4)  = [18800   19300   1000 2000];
    % Rockslides
        p1Slide(1,1:4) = [1       500     1    1000];

% Tlupana profile 1 feature coordinates
    % Sediment
        p2Sed(1,1:4)   = [1       500     1    1000];
    % Sills
        p2Sill(1,1:4)  = [2550  3050  1350  2350];
    % Rockslides
        p2Slide(1,1:4) = [7200  7700  3900 4900];
        p2Slide(2,1:4) = [15350 15850 3100 4100];

testSetCoords = vertcat(p1Sed,   p2Sed, ...
                        p1Sill,  p2Sill, ...
                        p1Slide, p2Slide);
```

```
function [trainingImages,                                ...
          p1SedImages, p1SillImages, p1SlideImages,      ...
          p2SedImages, p2SillImages, p2SlideImages]      ...
         = resizeImages(profile1, profile2,              ...
                        p1Sed, p1Sill, p1Slide,          ...
                        p2Sed, p2Sill, p2Slide,          ...
                        resizeRows, resizeColumns, resizeInterp)

%resizeImages
%
% The feature coordinates are used to extract sub-matrices from profiles
% resized to some standard number of rows and columns. Works for both the
% training and test image sets.
%
%   Usage example:
%   [trainingImages,                                ...
%    MU1SedImages, MU1SillImages, MUp1SlideImages, ...
%    MU2SedImages, MU2SillImages, MU2SlideImages]  ...
%   = resizeImages(muchalat1, muchalat2,           ...
%                  MU1Sed, MU1Sill, MU1Slide,      ...
%                  MU2Sed, MU2Sill, MU2Slide,      ...
%                  100, 100, 'nearest');
%
%   Resizes the 500x100 rectangles specified by selectTrainingSet output
%   to 100x100 images with nearest neighbor interpolation.
%
% Charles Garcia, 2015, cggarcia@uw.edu

% Profile 1 sediment training images
    for n = 1:size(p1Sed,1);
        p1SedImages{n} = ...
            imresize(profile1(p1Sed(n,3):p1Sed(n,4), ...
                              p1Sed(n,1):p1Sed(n,2)), ...
                              [resizeRows,resizeColumns], ...
                              resizeInterp);
    end

% Profile 1 sill training images
    for n = 1:size(p1Sill,1);
        p1SillImages{n} = ...
            imresize(profile1(p1Sill(n,3):p1Sill(n,4), ...
                              p1Sill(n,1):p1Sill(n,2)), ...
                              [resizeRows,resizeColumns], ...
                              resizeInterp);
    end

% Profile 1 rockslide training images
    for n = 1:size(p1Slide,1);
        p1SlideImages{n} = ...
            imresize(profile1(p1Slide(n,3):p1Slide(n,4), ...
                              p1Slide(n,1):p1Slide(n,2)), ...
                              [resizeRows,resizeColumns], ...
                              resizeInterp);
    end

% Muchalat profile 2 sediment training images
```

```
    for n = 1:size(p2Sed,1);
        p2SedImages{n} = ...
            imresize(profile2(p2Sed(n,3):p2Sed(n,4), ...
                              p2Sed(n,1):p2Sed(n,2)), ...
                              [resizeRows,resizeColumns], ...
                              resizeInterp);
    end

% Profile 2 sill training images
    for n = 1:size(p2Sill,1);
        p2SillImages{n} = ...
            imresize(profile2(p2Sill(n,3):p2Sill(n,4), ...
                              p2Sill(n,1):p2Sill(n,2)), ...
                              [resizeRows,resizeColumns], ...
                              resizeInterp);
    end

% Profile 2 rockslide training images
    for n = 1:size(p2Slide,1);
        p2SlideImages{n} = ...
            imresize(profile2(p2Slide(n,3):p2Slide(n,4), ...
                              p2Slide(n,1):p2Slide(n,2)), ...
                              [resizeRows,resizeColumns], ...
                              resizeInterp);
    end

trainingImages = horzcat(p1SedImages,   p2SedImages, ...
                         p1SillImages,  p2SillImages, ...
                         p1SlideImages, p2SlideImages);
```

```
function [eigenImages, eigenvalues, eigenvectors] =  ...
         computeEigenImages(trainingImages)

%computeEigenImages
%
% Computes the covariance matrix and associated eigenvectors and
% eigenvalues for a set of training images.
%
%   Usage example:
%   [eigenImages, eigenvalues, eigenvectors] =
%   computeEigenImages(trainingImages);
%
%   Computes the 42 eigenimages and eigenvalues associated with the
%   covariance matrix of the 42 images in trainingImages.
%
% Charles Garcia, 2015, cggarcia@uw.edu

    nImages = size(trainingImages,2);
    [rows, columns] = size(trainingImages{1});

% Reshape each training image into a vector and concatenate to form
% the full training set matrix.
    trainingMatrix = zeros(nImages, rows.*columns); % preallocate memory

    for n = 1:nImages
        trainingMatrix(n,:) = reshape(trainingImages{n}, 1, rows.*columns);
    end

    covarianceMatrix = trainingMatrix' * trainingMatrix;

    nEigs = nImages;  % number of eigenvector/eigenvalue pairs to compute
    sigmaEigs = 'lm'; % orders by eigenvalue from largest to smallest
    [eigenvectors, eigenvalues] = eigs(covarianceMatrix, nEigs, sigmaEigs);

% Reshape the eigenvectors into eigenimages with the same number of rows
% and columns as the training images.
    for n = 1:nImages
        eigenImages{n} = reshape(eigenvectors(1:rows.*columns, n), ...
                                 rows, columns);
    end

```

```
function [projectionImages, projectionCoefficients] =  ...
         projectImages(images, eigenvectors)

%projectImages
%
% Projects a set of images onto the training set eigenimages and returns
% the projected images and coefficients.
%
%   Usage example:
%   [projTestImages, projTestCoefficients] = ...
%   projectImages(testImages, eigenvectors);
%
% Charles Garcia, 2015, cggarcia@uw.edu

    nImages = size(images,2);
    [rows, columns] = size(images{1});

    for n = 1:nImages
        projectionCoefficients{n} =  ...
        reshape(images{n}, 1, rows.*columns)*eigenvectors;
    end

    for n = 1:nImages
        projectionImages{n} =  ...
        reshape(projectionCoefficients{n}*eigenvectors', rows, columns);
    end

```

Table 1. L^2 norms between the mean feature image projection coefficients (rows) and the test image projection coefficients (columns). Smaller norms indicate a better match between the test image and the feature type.

	1 Sediment	2 Sediment	3 Sill	4 Sill	5 Rockslide	6 Rockslide
Sediment	0.63	0.38	0.78	0.73	1.40	1.25
Sill	1.47	1.45	0.76	0.76	1.08	0.96
Rockslide	1.41	1.35	1.18	1.18	0.84	0.73

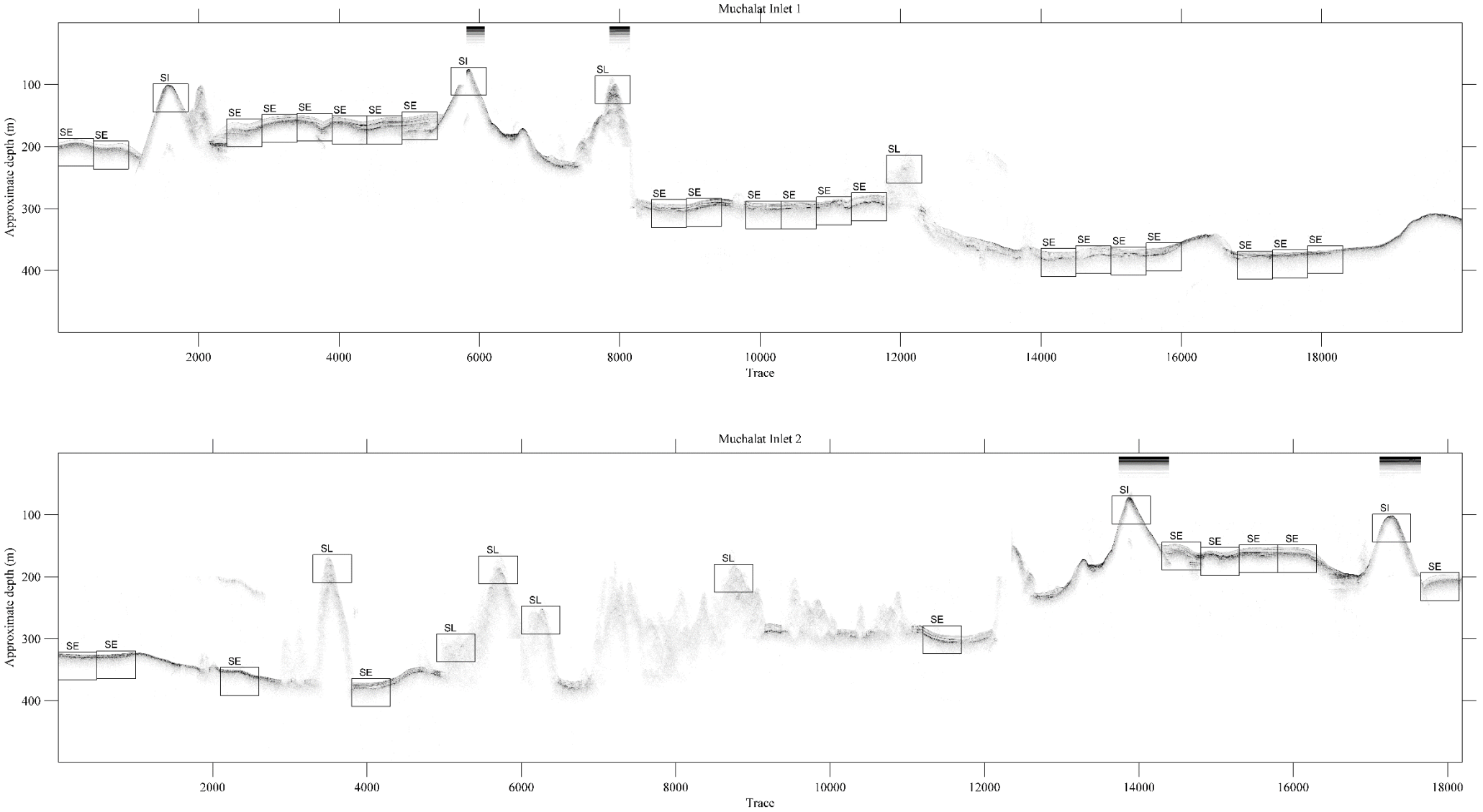

Figure 1. Aligned sub-bottom profiles from two surveys of Muchalat Inlet. The depth axis assumed a constant sound velocity of 1500 m s^{-1} in water. The rectangles indicate features in the training set of images. Sediment features are labeled SE, sill features are labeled SI, and rockslide features are labeled SL.

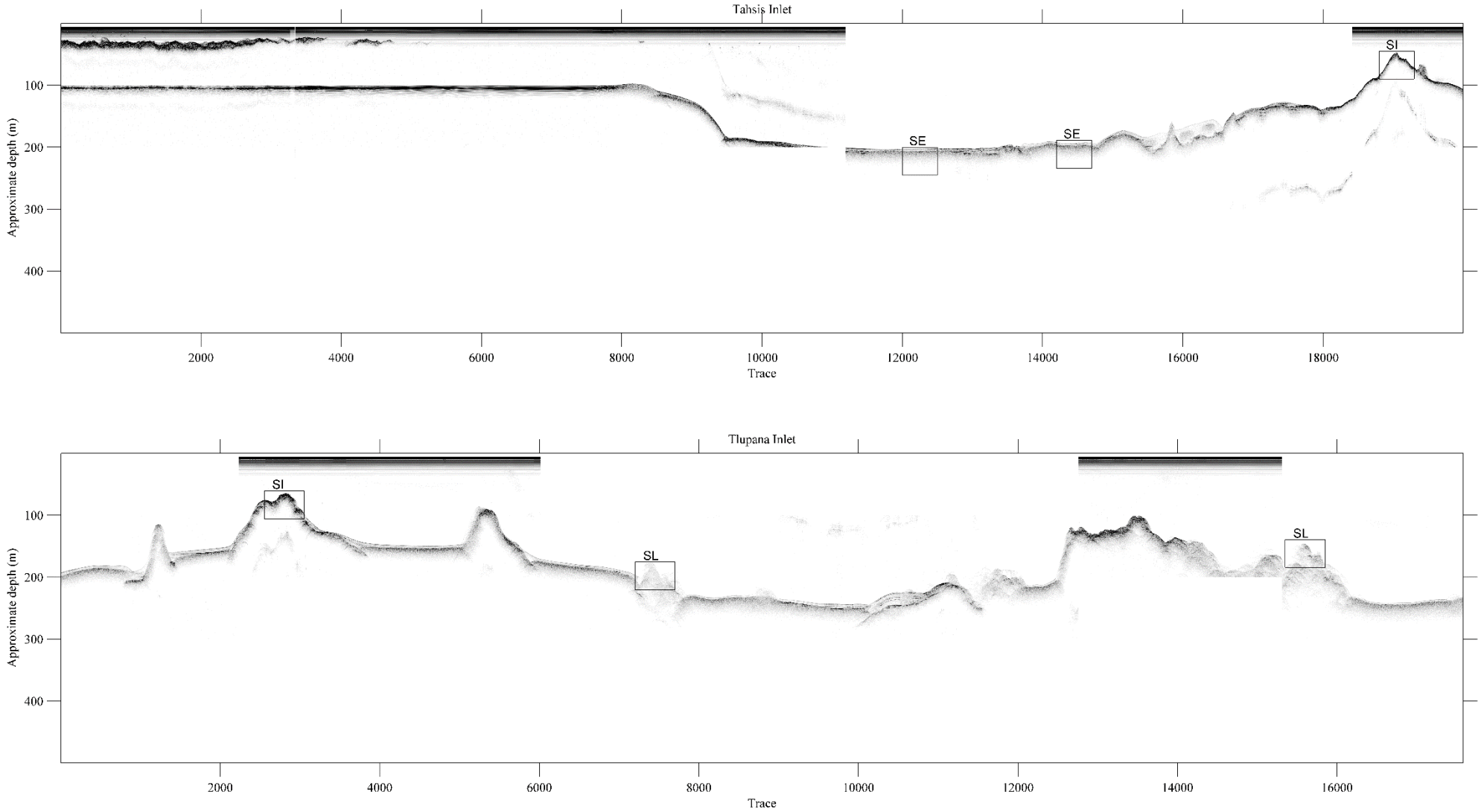

Figure 2. Aligned sub-bottom profiles from surveys of Tahsis and Tlupana Inlets. The depth axis assumed a constant sound velocity of 1500 m s^{-1} in water. The rectangles indicate features in the test set of images. Sediment features are labeled SE, sill features are labeled SI, and rockslide features are labeled SL.

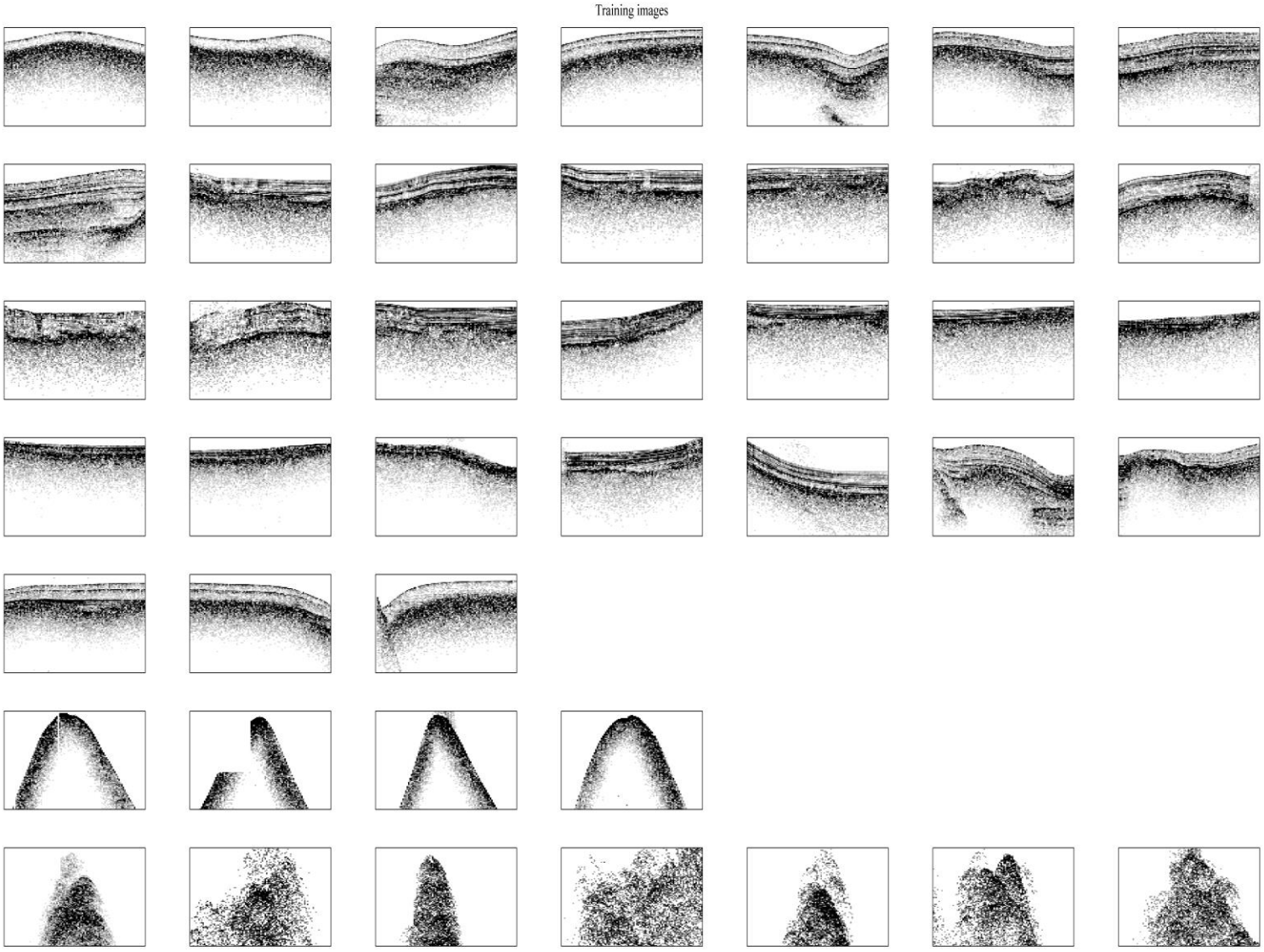

Figure 3. The set of training images from two surveys of Muchalat Inlet. The images in the first five rows are sediment features. The images in the next row are sills. The images in the last row are rockslides.

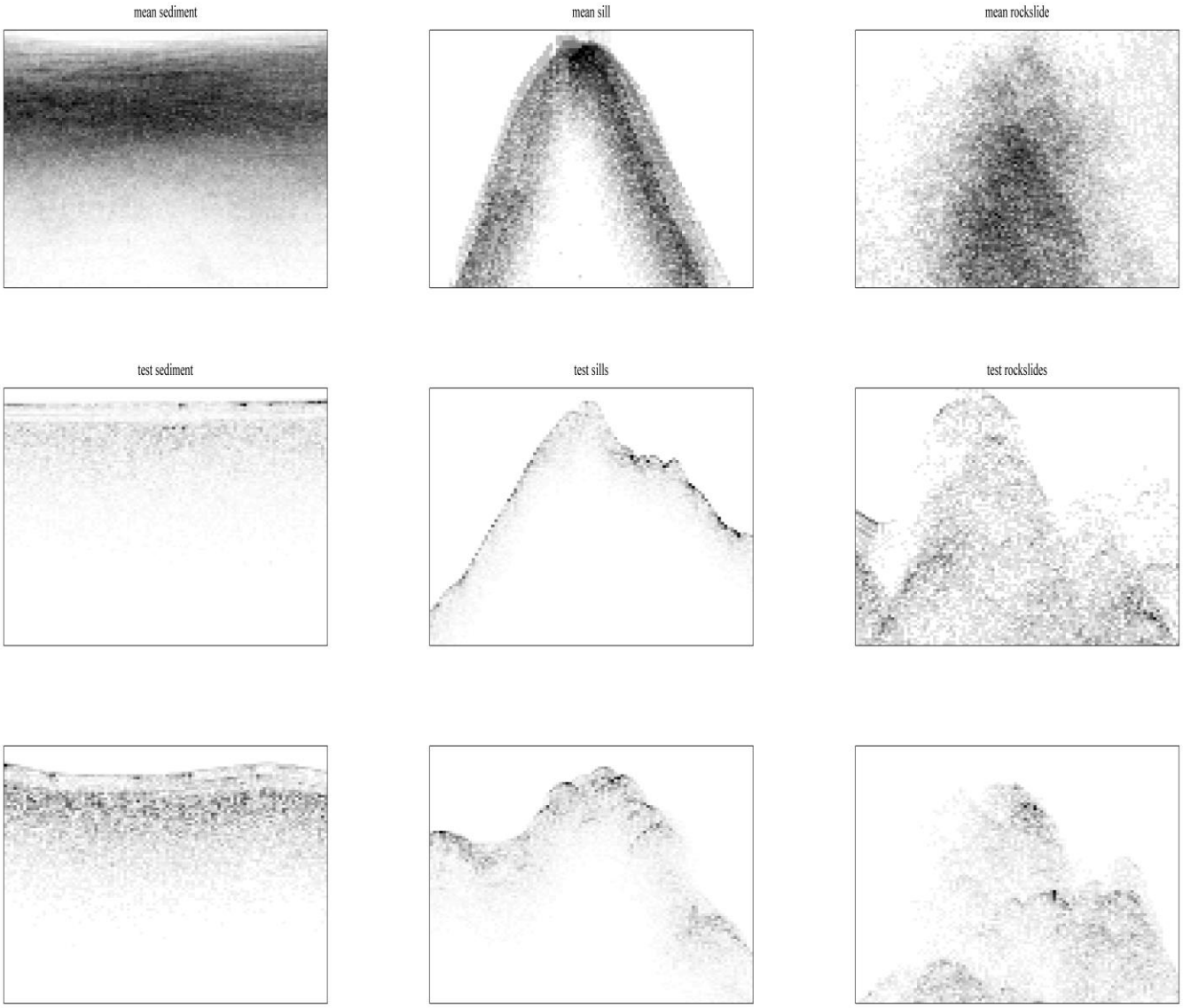

Figure 4. The mean feature images computed from the training set and the test images from surveys of Tahsis Inlet and Tlupana Inlet. The first row are mean feature images. The second and third row are test images.

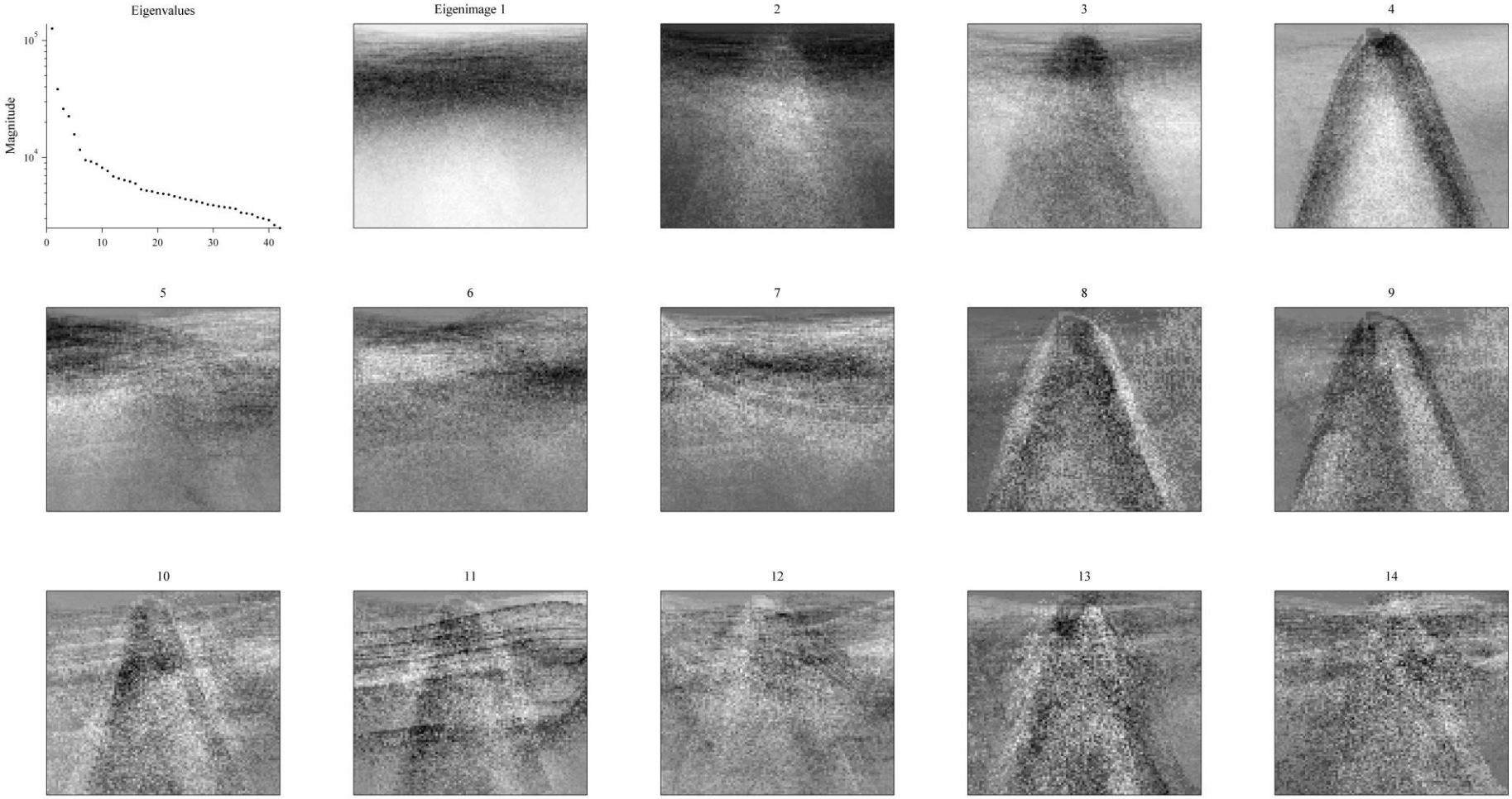

Figure 5. The eigenvalues (upper left) and eigenimages computed from the covariance matrix of the training images. The first 7 eigenvalues decrease rapidly corresponding to the high spatial variation between eigenimages 1 through 7. The rest of the eigenvalues decrease gradually and the rest of the eigenimages stagnate into a similar overall pattern with only minor variations.

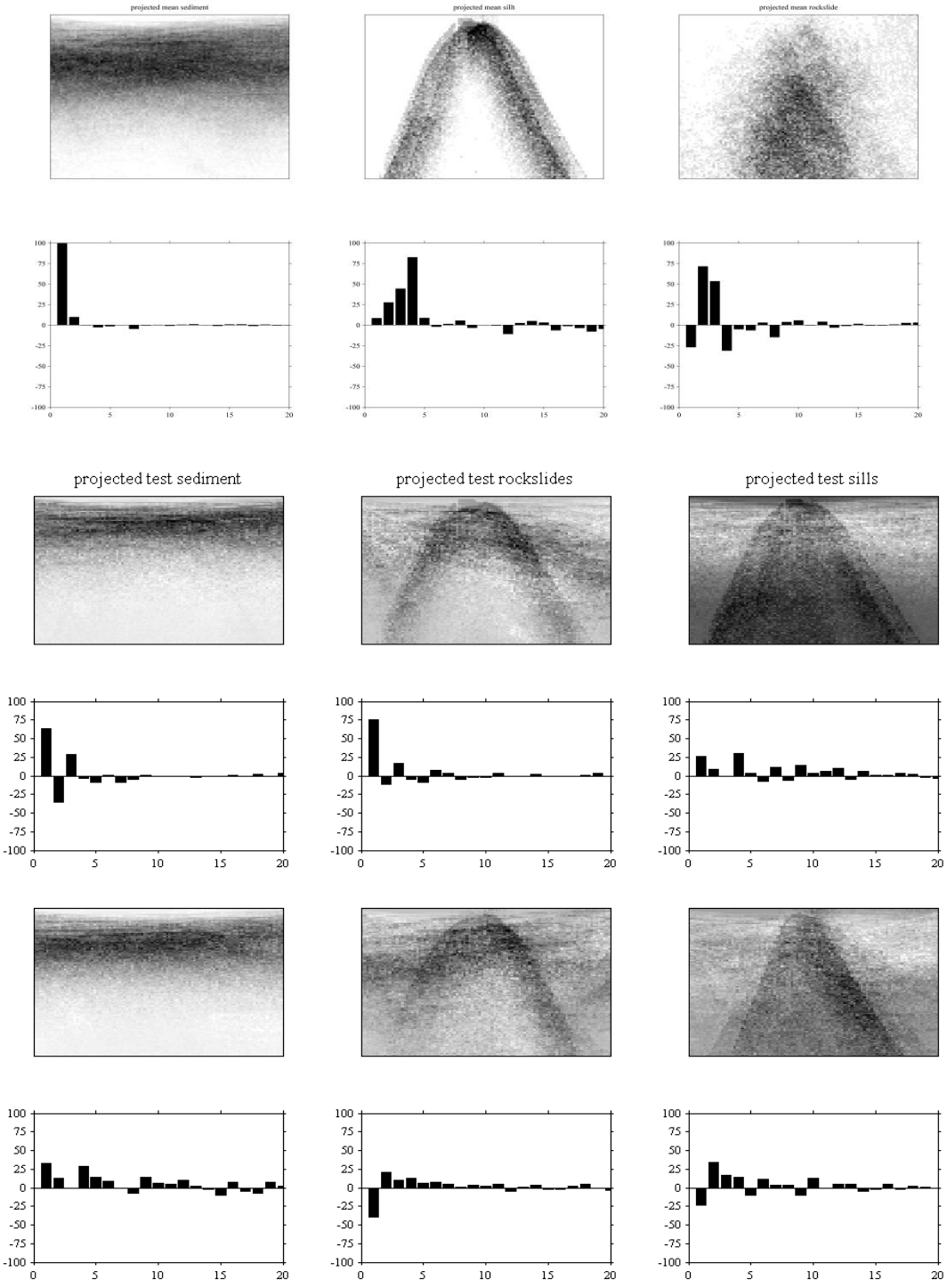

Figure 6. Projections of the mean feature images and test images with their projection coefficients immediately below.

Belhumeur, P.N., Hespanha, J.P., Kriegman, D., 1997. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. Pattern Analysis and Machine Intelligence, IEEE Transactions on 19, 711–720.

Delac, K., Grgic, M., Liatsis, P., 2005. Appearance-based statistical methods for face recognition, in: 47th International Symposium ELMAR, Zadar, Croatia. pp. 8–10.

Dove, D., Polyak, L., Coakley, B., 2014. Widespread, multi-source glacial erosion on the Chukchi margin, Arctic Ocean. Quaternary Science Reviews, APEX II: Arctic Palaeoclimate and its Extremes 92, 112–122. doi:10.1016/j.quascirev.2013.07.016

Gutowski, M., Bull, J., Henstock, T., Dix, J., Hogarth, P., Leighton, T., White, P., 2002. Chirp sub-bottom profiler source signature design and field testing. Marine Geophysical Researches 23, 481–492. doi:10.1023/B:MARI.0000018247.57117.0e

Kim, H.-J., Chang, J.-K., Jou, H.-T., Park, G.-T., Suk, B.-C., Kim, K.Y., 2002. Seabed classification from acoustic profiling data using the similarity index. The Journal of the Acoustical Society of America 111, 794–799. doi:10.1121/1.1433812

Kuhn, R., Junqua, J.-C., Nguyen, P., Niedzielski, N., 2000. Rapid speaker adaptation in eigenvoice space. IEEE Transactions on Speech and Audio Processing 8, 695–707. doi:10.1109/89.876308

Lee, G.H., Kim, H.J., Kim, D.C., Yi, B.Y., Nam, S.M., Khim, B.K., Lim, M.S., 2009. The acoustic diversity of the seabed based on the similarity index computed from Chirp seismic data. ICES J. Mar. Sci. 66, 227–236. doi:10.1093/icesjms/fsn142

Sirovich, L., Kirby, M., 1987. Low-dimensional procedure for the characterization of human faces. JOSA A 4, 519–524.

Turk, M.A., Pentland, A.P., 1991. Face recognition using eigenfaces, in: , IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91. Presented at the , IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91, pp. 586–591. doi:10.1109/CVPR.1991.139758

Viola, P., Jones, M.J., 2004. Robust Real-Time Face Detection. International Journal of Computer Vision 57, 137–154. doi:10.1023/B:VISI.0000013087.49260.fb

Windham, J.P., Abd-Allah, M.A., Reimann, D.A., Froelich, J.W., Haggar, A.M., 1988. Eigenimage filtering in MR imaging. J Comput Assist Tomogr 12, 1–9.

Yang, M.-H., Ahuja, N., Kriegman, D., 2000. Face recognition using kernel eigenfaces, in: 2000 International Conference on Image Processing, 2000. Proceedings. Presented at the 2000 International Conference on Image Processing, 2000. Proceedings, pp. 37–40 vol.1. doi:10.1109/ICIP.2000.900886