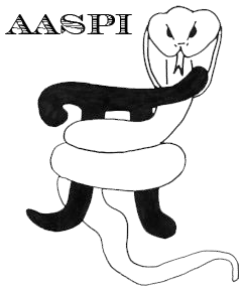


VOLUMETRIC SELF-ORGANIZING MAPS FOR 3D SEISMIC FACIES ANALYSIS – PROGRAM som3D

Contents

Computation flow chart

This self-organizing map (SOM) 3D facies analysis program is a tool to generate a seismic facies map from multiple seismic attributes in an unsupervised fashion. Different exploration target or seismic facies may be sensitive to different seismic attributes, and often times interpreters need to use multiple attributes to delineate features within the area of interest. Taking multiple attribute inputs, SOM tries to generate a facies map that captures most, if not all variations in the input attributes. We consider the SOM process as a projection from multidimensional attribute space to a 2D space, and the **som3d** program will output two files of projections on two SOM axes, which can be directly crossplotted in **crossplot** or other modern interpretation packages using a 2D RGB colorbar, making visualization more convenient and interactive. Below is the flowchart showing the workflow of 3D seismic facies analysis (see next page).

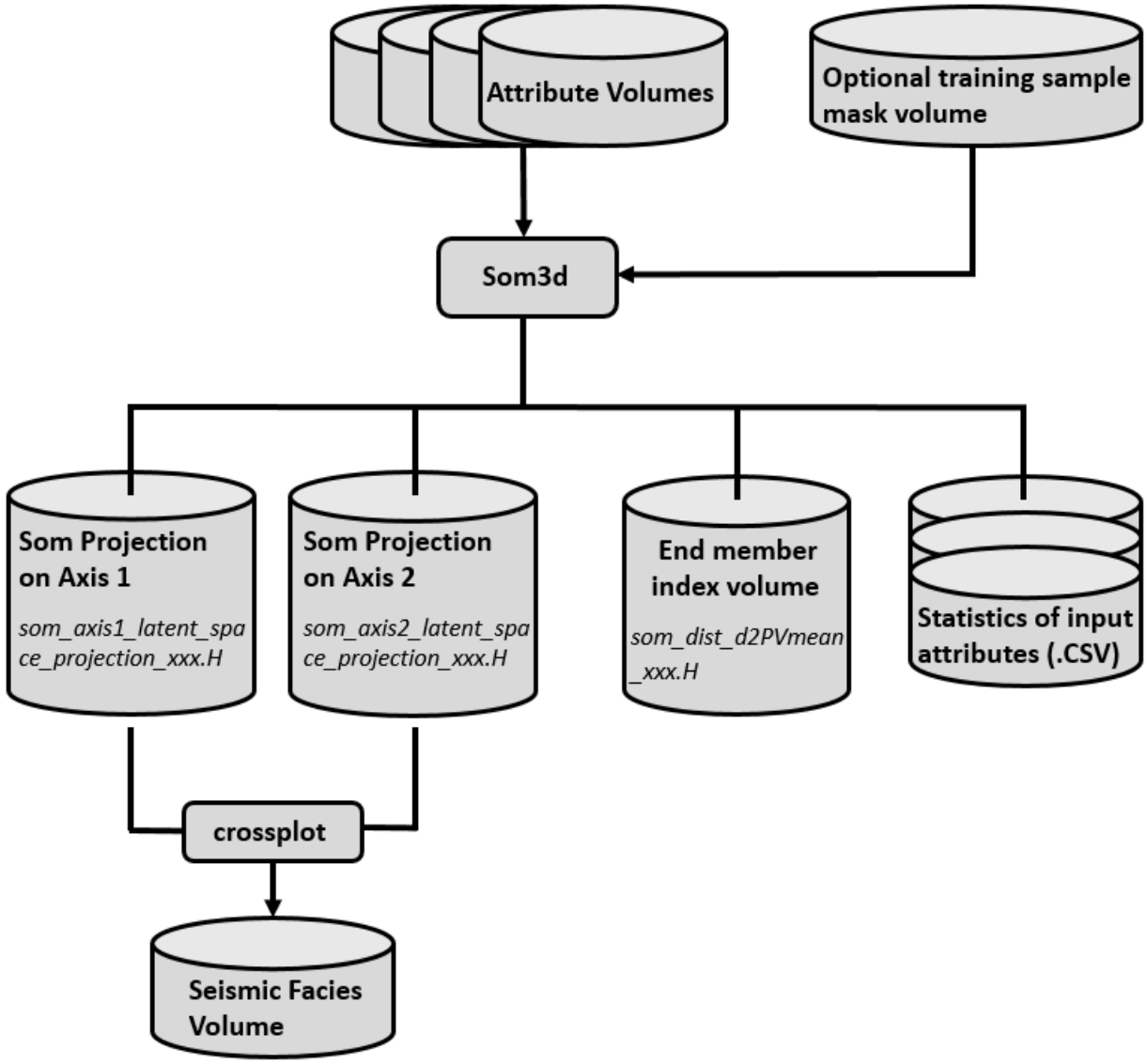


Figure 1.

Output file naming convention

Program **som3d** will generate the following files that provide statistics on the input multiattribute data, where files ending in *.csv (comma-separated values) can be plotted using Microsoft Excel software:

Output file description	File name syntax
Attribute mean values	som3d_mean_*unique_project_name_suffix*.csv
Attribute standard deviations	som3d_std *unique_project_name_suffix*.csv
Multiattribute covariance matrix	som3d_covariance_*unique_project_name_suffix*.csv
Multiattribute eigenvectors	som3d_eigenvectors_*unique_project_name_suffix*.csv
Multiattribute eigenvalues	som3d_eigenvalues_*unique_project_name_suffix*.csv

Program **som3d** will also generate the following output files:

Output file description	File name syntax
Program log information	som3d_*unique_project_name_suffix*.log
Program error/completion information	som3d_ *unique_project_name_suffix*.err
Classified data	som3d_cluster_number_*unique_project_name_suffix*.H
Classified data projected on SOM axis 1	som3d_projection_axis1_ *unique_project_name_suffix*.H
Classified data projected onto SOM axis 2	som3d_projection_axis2_ *unique_project_name_suffix*.H
Distance of each data vector to its cluster center	som3d_distance_*unique_project_name_suffix*.H
Prototype vectors	som3d_prototype_vector_waveform_*unique_project_name_suffix*.H

where the values in red are defined by the program GUI. The errors we anticipated will be written to the *.err file and be displayed in a pop-up window upon program termination. These errors, much of the input information, a description of intermediate variables, and any software trace-back errors will be contained in the *.log file.

SOM classification is initialized using the first two eigenvalues and eigenvectors, and in this application are identical to those generated by program **pca3d**. This 2D plane (the simplest manifold in *N*-dimensional attribute space) is sampled by a suite of regularly spaced prototype

vectors which are then projected onto the SOM latent space. At each iteration, the location of each prototype vector moves in the *N*-dimensional space to better represent the training data. These prototype vectors (some workers call them "neurons") are then projected onto the 2D latent space at each iteration. Each sample in the input data represents a time slice, phantom horizon slice, or stratal slice. In order to classify, the input data are scaled using the mean and standard deviation for each slice. For this reason, there are two versions of the prototype vectors – the one that is scaled and used internal to the program, and the one that is unscaled (in "world coordinates") and may be more useful to an interpreter. Both of these vectors can be plotted against a color map called the *prototype vector color matrix*. The classified results are provided in two formats – as a labeled data volume (consisting of integer values stored as floating point numbers) that can be plotted against a corresponding classification color bar, or as the classes projected against SOM latent space axes 1 and 2, which can be plotted using aaspi_crossplot or crossplot tools available in commercial software. Most commercial software packages allow an interpreter to define polygons in the crossplot space, thereby providing more control in constructing seismic facies.

As with programs **rgb_cmy_plot**, **crossplot**, and **hlsplot**, the user can request the following optional colorbars for the more common interpretation software packages:

Output file description	File name syntax
Petrel classification color bars	som_waveforms_colors_*unique_project_name_suffix*.iesx
Landmark classification color bars	som_waveforms_colors_*unique_project_name_suffix*.cl2
Kingdom Suite classification color bars	som_waveforms_colors_*unique_project_name_suffix*.CLM
Seisware classification color bars	som_waveforms_colors_*unique_project_name_suffix*.xml
Voxelgeo classification color bars	som_waveforms_colors_*unique_project_name_suffix*.color
Geoprobe classification color bars	som_waveforms_colors_*unique_project_name_suffix*.gpc
Transform classification color bars	som_waveforms_colors_*unique_project_name_suffix*.cmp
Geomodeling classification color bars	som_waveforms_colors_*unique_project_name_suffix*.geomodeling
Seisware classification color bars	som_waveforms_colors_*unique_project_name_suffix*.CLM

Because the AASPI software uses the Petrel *.alut format files for its display; this file will always be generated.

Theory

Self-organizing map (SOM) is closely related to vector quantization methods (Haykin, 1999). Initially we assume that the input are represented by *J* vectors in a *N*-dimensional vector space R_n, $\mathbf{x}_j = [\mathbf{x}_{j1}, \mathbf{x}_{j2}, \mathbf{x}_{j3} \ldots \mathbf{x}_{jN}]$ where *N* is the number of input attributes (or amplitude samples for "waveform" classification) and *j=1,2,…,J* is the number of vectors analyzed. The objective of the algorithm is to organize the dataset of input seismic attributes into a geometric structure called the SOM. SOM consists of neurons or prototype vectors (PVs) organized by a lower-dimension grid, usually 2D, which are representative of the input data that lies in the same N-dimensional space as the input seismic attributes. PVs are also termed as SOM units and typically arranged in 2D hexagonal or rectangular structure maps that preserve the neighborhood relationship among the PVs. In this manner PVs close to each other are associated with input seismic attribute vectors that are similar to each other. The number of these PVs in the 2D map determines the effectiveness and generalization of the algorithm. Let's consider a 2D SOM represented by P prototype vectors $\mathbf{m}_i$, $\mathbf{m}_i = [\mathbf{m}_{i1}, \mathbf{m}_{i2} \ldots \mathbf{m}_{iN}]$, where *i=1, 2, …, P* and *N* is the dimension of these vectors defined by the number of input attributes (or samples for waveform classification).

During the SOM training process, an input vector is initialized and is compared with all *N*-dimensional PVs on the 2D grid, or latent space. The prototype vector with the best match (the winning PV) will be updated as a part of SOM neighborhood training.

Given this background, Kohonen (2001) defines the SOM training algorithm using the following five steps:

Step 1: Consider an input vector, which is randomly chosen from the set of input vectors.

Step 2: Compute the Euclidean distance between this vector x and all PVs $\mathbf{m}_i$, i=1, 2,…p. The prototype vector $\mathbf{m}_b$, which has the minimum distance to the input vector x, is defined to be the "winner" or the Best Matching Unit, $\mathbf{m}_b$:

$$\left|\left|\mathbf{x} - \mathbf{m}_b\right|\right| = \mathrm{MIN}\left\{\left|\left|\mathbf{x} - \mathbf{m}_i\right|\right|\right\} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Step 3: Update the "winner" prototype vector and its neighbors. The updating rule for the weight of the i^{th} PV inside and outside the neighborhood radius $\sigma(t)$ is given by:

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t) h_{bi}(t)[\mathbf{x} - \mathbf{m}_i(t)] \quad \text{if } \left|\left|\mathbf{r}_i - \mathbf{r}_b\right|\right| \leq \sigma(t) \quad (2a)$$

$$= \mathbf{m}_i(t) \quad \text{if } \left|\left|\mathbf{r}_i - \mathbf{r}_b\right|\right| > \sigma(t), \quad (2b)$$

where the neighborhood radius defined as $\sigma(t)$ is predefined for a problem and decreases with each iteration t. $\mathbf{r}_b$ and $\mathbf{r}_i$ are the position vectors of the winner PV $\mathbf{m}_b$ and the ith PV m_i respectively. We also

define $h_{bi}(t)$ as the neighborhood function, $\alpha(t)$ as the exponential learning function and T as the length of training. $h_{bi}(t)$ and $\alpha(t)$ decrease with each iteration in the learning process and they are defined as

$$h_{bi}(t) = e^{-(||\mathbf{r}_b - \mathbf{r}_i||^2/2\sigma^2(t)} \quad \text{, and} \quad \text{..... (3)}$$

$$\alpha(t) = \alpha_0\left(\frac{0.005}{\alpha_0}\right)^{t/T} \quad . \quad \text{..... (4)}$$

Step 4: Iterate through each learning step (steps 1-3) until the convergence criterion (which depends on the predefined lowest neighborhood radius and the minimum distance between the PVs in the latent space) is reached.

Step 5: Color-code the trained PVs using 2D or 3D gradational colors (Matos et al. 2009). We will use an HSV model, for 2D spaces will be defined as hue, $\mathcal{H}$,

$$\mathcal{H} = \tan^{-1}\left(\frac{v - 1/2}{u - 1/2}\right) \quad \text{..... (5)}$$

and saturation, $\mathcal{S}$, as

$$\mathcal{S} = [(u - 1/2)^2 + (v - 1/2)^2]^{1/2} \quad \text{..... (6)}$$

where u and v are the projected components onto the 2D latent space defined by the eigenvectors $\mathbf{v}^{(1)}$ and $\mathbf{v}^{(2)}$. The new sets of PVs are colored using the 2D HSV color palette with equations 5 and 6.

In traditional Kohonen SOM, the position of an SOM node in the SOM latent space is only based on the distance between the corresponding prototype vector (the projection of an SOM node in the input data space) and the nearest data vector in the input space. In our implementation, we add a step of adjusting the position of all SOM nodes according to their distances from the current winning node (best matching unit) in both input data space and SOM latent space. The adjustment rule is (Shao and Yang, 2012):

$$\mathbf{r}_k(t+1) = \mathbf{r}_k(t) + \alpha(t) \cdot \left(1 - \frac{\delta_{vk}}{d_{vk}}\right) \cdot \left(\mathbf{r}_v(t) - \mathbf{r}_k(t)\right), \ \forall\, k \neq v. \quad \text{.....(7)}$$

In Equation 7, $\mathbf{r}_k(t)$ is the positon of an SOM node before adjustment; $\mathbf{r}_k(t+1)$ is the position of an SOM node after adjustment; $\mathbf{r}_v(t)$ is the position of the current winning node; δ_{vk} and d_{vk} are the distances between an SOM node and the current winning node in input data space and in the SOM latent space, respectively. $\alpha(t)$ is the learning rate which exponentially decays over iterations.

The input of our SOM3D algorithm consists of several mathematically independent volumetric attributes where the number of input attributes determines the mathematical dimensionality of the data. Due to the limitation of our visualization software which provides only 256 colors, we have limited our over-defined prototype vectors to a maximum of *J*=256. In this application, we normalize our input data vectors using a Z-score algorithm. Thus our input data has a vector assigned to each of the (*x*, *y*, *z*) locations in our volume (which

are actually the normalized input attribute values at that location).

We name this new volume the normalized multi-attribute volume and project it onto a 2D latent space using Principal Component Analysis. The 2D latent space is defined as explained earlier. If there are six input attribute volumes, each of the PVs in the 2D latent space is 6-dimensional. This 2D latent space is sampled uniformly by 256 PVs. The PVs are trained in the 2D latent space and their positions updated after each iteration, resulting in the new updated position of the PVs. When the updating slows down, the training process stops. With an increasing number of iterations, the PVs move closer to each other and to the data points within the latent space. The HSV colors are assigned to the PVs according to their distance from their center of mass and their azimuth (equations 5 and 6). Once trained, the distance is computed between each PV, $\mathbf{m}_i'$, and the multiattribute data vector, x, at each voxel using

$$\left|\left|\mathbf{x}-\mathbf{m}_\mathbf{b}'\right|\right| = \min\left\{\left|\left|\mathbf{x}-\mathbf{m}_i'\right|\right|\right\} \quad \text{..... (8)}$$

where $\mathbf{m}_b'$ is the nearest PV to the input data sample vector $\mathbf{x}$. Each voxel is then assigned the color of $\mathbf{m}_b'$. In this manner, two dissimilar neighboring samples in the seismic volume will be far apart in the latent space and have different colors. Conversely, two similar samples in the seismic volume will have nearly the same color. Each color represents a seismic facies, most of which are geologic facies, but some which may be seismic 'noise' facies.

Users also have the option to weigh input attributes differently in a data-adaptive fashion. The weight matrix **W** is defined as a function of interpreter's knowledge and attributes' contribution to SOM. Inspired by Benabdeslem and Lebbah (2007), given *N* input attributes and *P* prototype vectors, we define ω_i, the i^{th} attribute's contribution to an SOM model, as:

$$\omega_i = \sum_{j=1}^{P} d_j \frac{|m_{ji}|}{\sum_{k=1}^{N}|m_{jk}|}, \quad \text{..... (9)}$$

and

$$d_j = \frac{h_j}{J}, \quad \text{..... (10)}$$

where h_j is the number of multiattribute training samples that are nearest to the j^{th} prototype vector, J is the total number of multiattribute training samples, d_j represents the density of training samples assigned to the j^{th} prototype vectors, and p_{jk} is the value of the j^{th} prototype vector along dimension k (the dimension of the k^{th} attribute). Physically, if a prototype vector has a very large value in the dimension of the target attribute, and a large percentage of training samples are close to this prototype vector, then the target attribute's contribution at this prototype vector is significant. Summing up over all the prototype vectors, we then arrive at the target attribute's contribution to the whole SOM model. To ensure an overall optimal contribution measurement for a given group of attributes, we propose to use an exhaustive search over all combinations of three or more attributes,

$$S = \sum_{i=3}^{N} \frac{N!}{i!(N-i)!}, \quad \text{..... (11)}$$

and then weight by the number of attributes in each combination and take the sum over all used

combinations:

$$\omega_i = \sum_{l=1}^{S} N_l \widetilde{\omega}_{il}, \quad \text{(12)}$$

where ! denotes the factorial operation, *S* is the total number of SOM models to be searched, N_l is the number of attributes in the l^{th} combination, $\widetilde{\omega}_{il}$ is the contribution of the i^{th} attribute to the l^{th} SOM model, and ω_i is the final contribution of the i^{th} attribute to the SOM. Although the method involves running SOM multiple times with different input attribute combinations, it is an embarrassingly parallel problem so that the increase in computation time over the traditional SOM is negligible given sufficient amount of threads/processors.

We use skewness and kurtosis to quantify the distribution of an attribute, and we weigh more on attributes that exhibit smaller absolute skewness and higher kurtosis. Skewness, which is the third moment of the standard score of a variable **x**, is defined as:

$$s(\boldsymbol{x}) = \mathbb{E}\left[\left(\frac{\mathbf{x}-\bar{\mathbf{x}}}{\sigma_x}\right)^3\right], \quad \text{(13)}$$

where $\bar{x}$ is the mean of variable x, σ_x is the standard deviation, and $\mathbb{E}$ represents expectation. Similarly, kurtosis is the fourth moment of the standard score of a variable **x** and is defined as:

$$k(\mathbf{x}) = \mathbb{E}\left[\left(\frac{\mathbf{x}-\bar{\mathbf{x}}}{\sigma_x}\right)^4\right]. \quad \text{(14)}$$

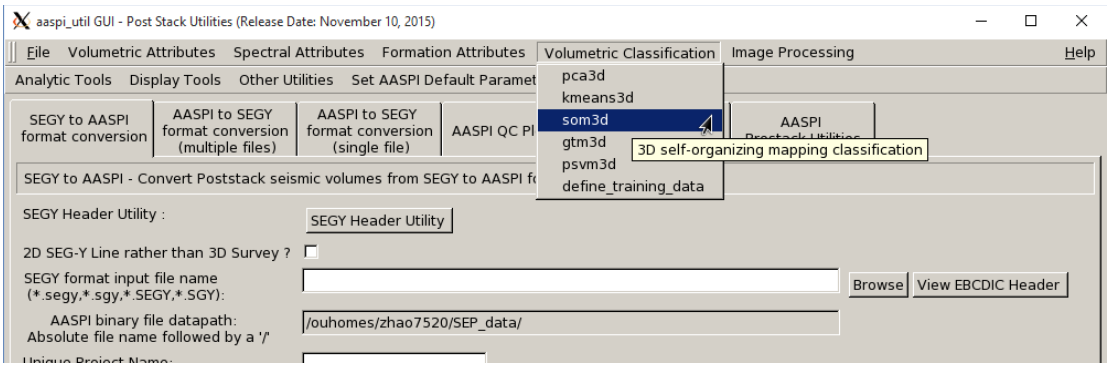
In practice, the skewness and kurtosis are precomputed before determining the attribute contribution **ω**. After the computation of **ω**, we further normalize both skewness and kurtosis to range between zero and one. Weighting the previously defined **ω** using skewness and kurtosis, and normalize again using the z-score:

$$w_i = \left(3 - \frac{|s_i| - \min\limits_{i=1,N}|s_i|}{\max\limits_{i=1,N}|s_i| - \min\limits_{i=1,N}|s_i|} - \frac{k_i - \min\limits_{i=1,N} k_i}{\max\limits_{i=1,N} k_i - \min\limits_{i=1,N} k_i}\right)\omega_i, \quad \text{(15)}$$

$$\widehat{w}_i = \frac{w_i - \bar{w}}{\sigma_w}. \quad \text{(16)}$$

Here, w_i is the weight of attribute *i* before z-score normalization, $\bar{w}$ is the mean of w_i, σ_w is the standard deviation, and $\widehat{w}_i$ is the weight of attribute *i* after the z-score. In equation 15, because the skewness term and kurtosis term are both normalized to range between zero and one, we assume an equal impact of skewness and kurtosis. At the same time, the absolute value of w_i is of less interest, as we further normalize it to be $\widehat{w}_i$ using z-score. Finally, we constrain the weight to range from zero to two using a sigmoid function, and defining the elements of the diagonal weight matrix **W** to be:

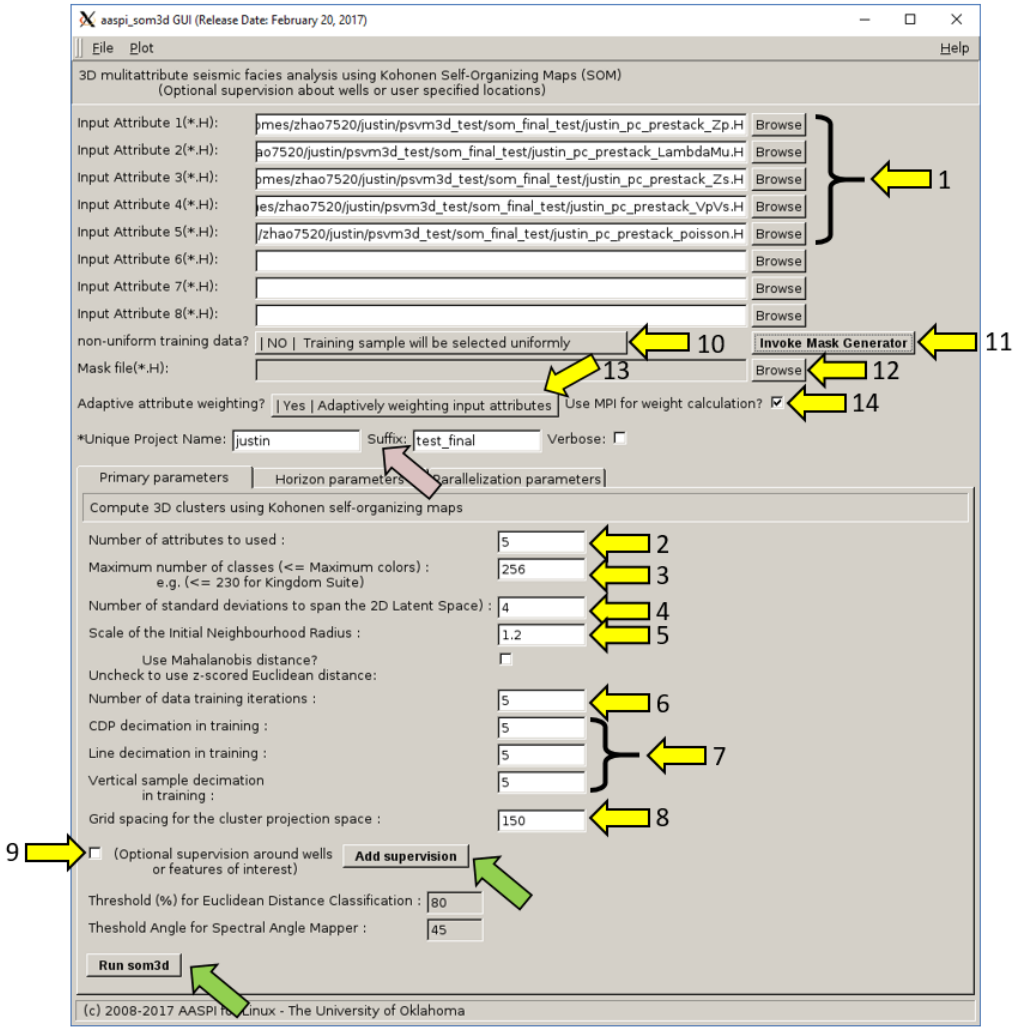
$$W_{ii} = \frac{2}{1+e^{-\widehat{w}_i}}. \quad \text{(17)}$$



This Program **som3d** is launched from the *3D facies classification* in the main **aaspi_util** GUI

Computing som3d module

Setting the primary parameters is the first step of the analysis. Use the browser on the first eight lines to choose the input seismic data file (*Arrow 1*). It is not mandatory to take in eight inputs. The number of inputs can vary from two – eight. The input attributes that one considers for facies analysis will vary according to the requirements. For identifying the depositional facies variation the volumetric attributes such as dip magnitude, coherency, GLCM attributes, spectral magnitude, coherent energy can be considered as input. For characterizing geo-mechanical variation in shale plays, one should consider different volumes that helps in identifying the rock physics such as inversion volumes, lambda-rho, mu-rho, intercept or gradient AVO volumes, etc. Specify the number of input attributes in the field labeled "Number of attributes to use" (*Arrow 2*). This value will be updated automatically when a file is selected. Do not forget to give a "*Unique Project Name*". A Z-score algorithm is used to normalize the input files.



The maximum *number of classes* can be any large number (*Arrow 3*). In using SOM, we always start with an over-defined number of classes and allow the algorithm to automatically form fewer classes. Most of the commercial visualization software can only display 256 colors thus generally is <= 256. However, with a more uniform sampling of the latent space, we generally have more confidence in clustering. The eigenvectors and the eigenvalues are now calculated internally. They serve as the first approximation to the latent space forming the initial set of untrained vectors. The standard deviation value scales the 2D Latent space (*Arrow 4*). A value of 3σ makes the latent space represent 97% of the data. Set the initial value of the SOM neighborhood radius within which all neighbor prototype vectors are updated (*Arrow 5*). Put the maximum *number of iterations* (*Arrow 6*). Select the decimation rate of input data used for training (*Arrow 7*). The operation window options are defined in the **Operation Window** tab shown below. In a 2017 update, we add in the option to use a user-defined mask file in defining the otherwise uniformly generated training samples (*Arrow 10* and *11*). We demonstrate this function in a later section named "**use non-uniform training data"**. Another unique option is to use data-adaptively derived weights to emphasize (and deemphasize) each input attribute (*Arrow 13*). If selected, input attributes will be weighted using weights internally computed

from the **som3d** program. The mathematical details on weight calculation are provided in the **Theory** section earlier in this documentation. During this input attribute weight evaluation process, the program builds multiple SOM models in a parallel fashion. If a user is using a computer with limited memory, the program provides an option for sequential mode in weight calculation by unchecking the box at *Arrow 14*.

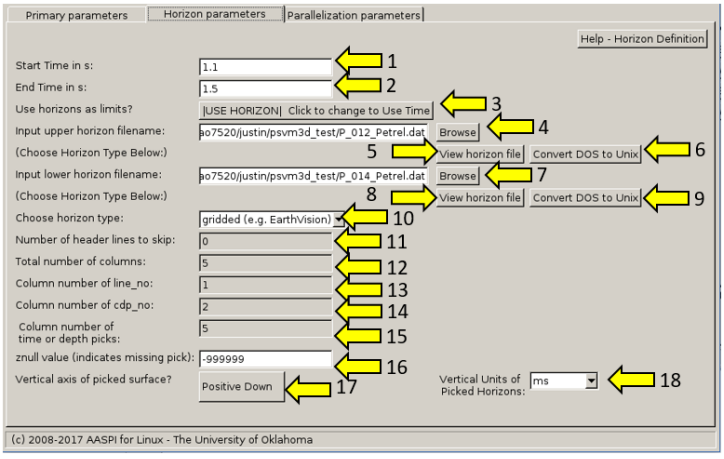
Use non-uniform training data

By default, the SOM program decimates input attribute volumes uniformly along line, cdp, and time (depth) to extract training data that best represent the whole input data. This is the best way to explore the dominant natural clusters within the whole survey. If the user prefers to limit the SOM analysis to a smaller region within the seismic survey, or opts to emphasize more on variations found in a smaller region versus the general trend within the survey, he/she has the option to use non-uniformly decimated data to build the SOM model. Click *Arrow 10* button to toggle between uniformly sampled and non-uniformly sampled training data. If using non-uniformly sampled training data, the user needs to provide an AASPI .H format mask file to define the training sample locations. Such mask file can be generated using the utility **Mask Generator** (*Arrow 11*), which can also be invoked form the AASPI main window as **Volumetric Classification/define_training_data**. A separate documentation is provided for this module.

Depending on how much emphasis an interpreter wants to put on a particular region of interest, a mask file may contain 1. Uniformly sampled background only, 2. Equally weighted background and picked region(s), 3. Background and picked region(s) with varied weights, and 4. Picked region(s) only. Users are encouraged to experiment with different mask files to see how facies change with alternative training sample extraction schemes.

Define the operation window

A user has the option to use either a constant time window or a window defined by top and bottom horizons. The functionalities in defining the operation window are discussed below.



Horizon definition

The horizon definition panel will look the same for almost all AASPI GUIs:

1. Start time (upper boundary) of the analysis window.
2. End time (lower boundary of the analysis window.
3. Toggle that allows one to do the analysis between the top and bottom time slices described in 1 and 2 above, or alternatively between two imported horizons. If *USE HORIZON* is selected, all horizon related options will be enabled. If the horizons extend beyond the window limits defined in 1 and 2, the analysis window will be clipped.
4. Browse button to select the name of the upper (shallower) horizon.
5. Button that displays the horizon contents (see Figure 1).
6. Button to convert horizons from Windows to Linux format. If the files are generated from Windows based software (e.g. Petrel), they will have the annoying carriage return (^M) at the end of each line (Shown in Figure 1). Use these two buttons to delete those carriage returns. Note: This function depends on your Linux environment. If you do not have the program **dos2unix** it may not work. In these situations, the files may have been automatically converted to Linux and thus be properly read in.
7. Browse button to select the name of the lower (deeper) horizon.
8. Button that displays the horizon contents (see Figure 1).
9. Button to convert horizons from Windows to Linux format (see 6 above).
10. Toggle that selects the horizon format. Currently *gridded* (e.g. EarthVision in Petrel) and *interpolated* (ASCII free format, e.g. SeisX) formats are supported. The gridded horizon are nodes of B-splines used in mapping and have no direct correlation to the seismic data survey. For example, gridded horizons may be computed simply from well tops. The x and y locations are aligned along north and east axes. In contrast interpolated horizons are defined by *line_no, cdp_no (crossline_no)* and *time* triplets for each trace location. Examples of both formats are shown in Figure 1. If *interpolated* is selected, the user needs to manually define each column in the file.
11. Number of header lines to skip in the *interpolated* horizon files.
12. Total number of columns in the *interpolated* horizon files.
13. Enter the column number containing the *line_no (inline_no)* of the interpolated data triplet.
14. Enter the column number containing the *cdp_no (crossline_no)* of the interpolated data triplet.
15. Enter the column number containing the *time* or *depth* value of the interpolated data triplet.
16. *Znull* value (indicate missing picks) in the horizon files.
17. Toggle to choose between *Positive Down* and *Negative Down* for the horizon files (e.g. Petrel uses negative down).
18. Choose the vertical units used to define the horizon files (either *s*, ms, *kft, ft, km,* or *m*).

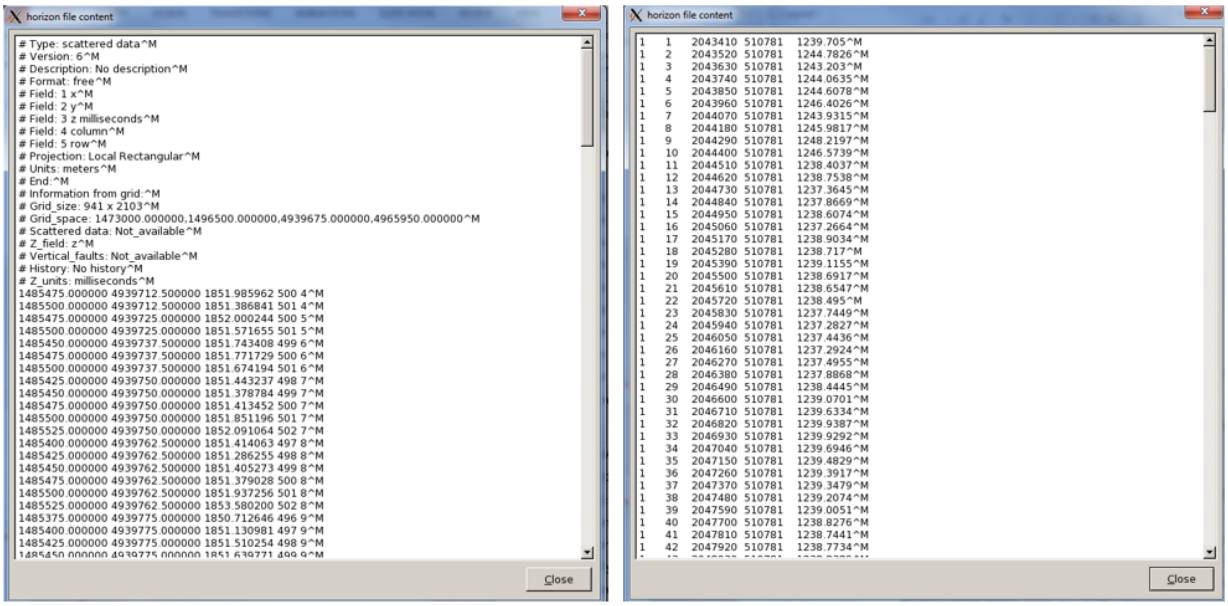
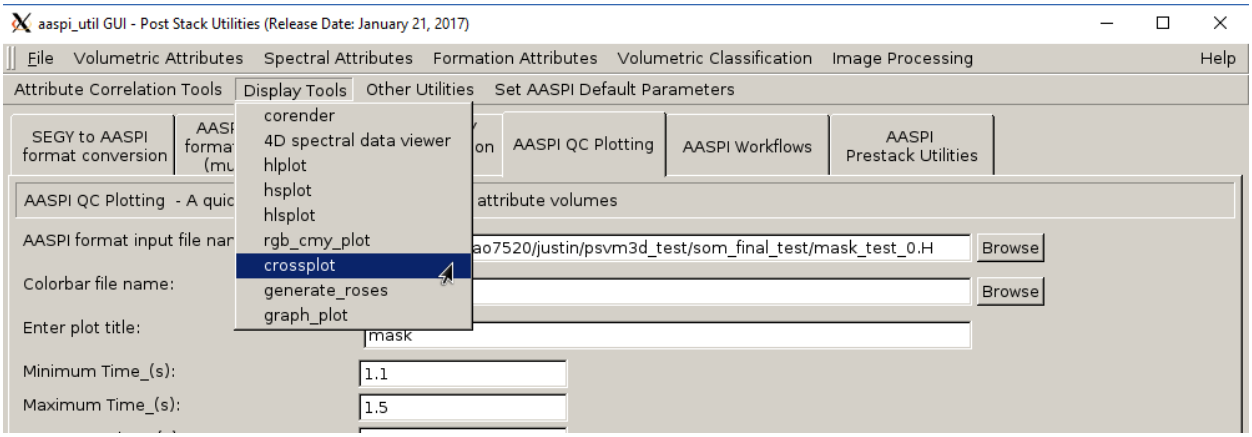


Figure 2**.** (left) A gridded horizon file (EarthVision format). (right) An interpolated horizon file with five columns (ASCII free format).

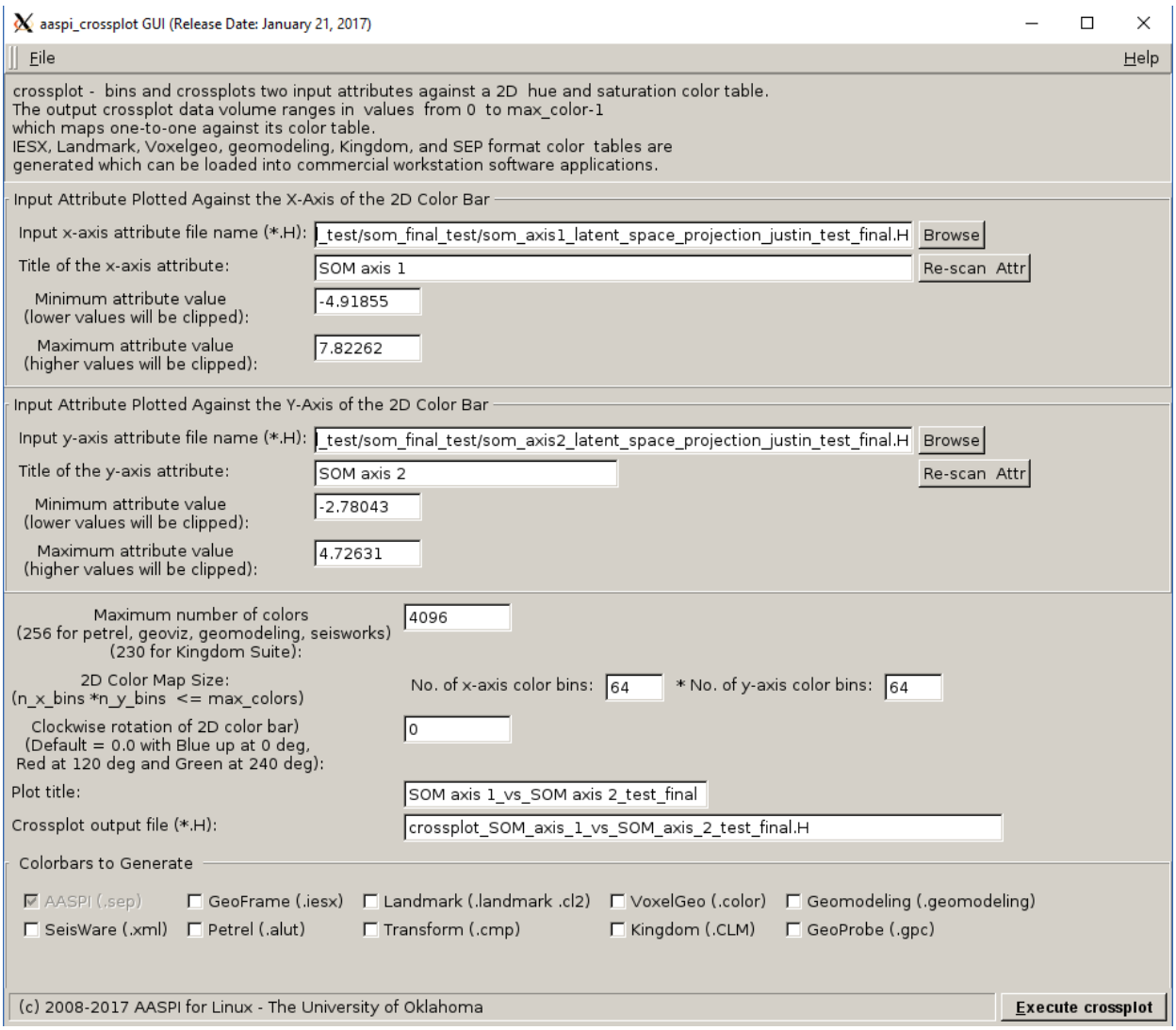
Displaying the results

Plotting the SOM facies using aaspi_crossplot

The user can use **crossplot** module in the **aaspi_util** to crossplot two SOM axes in order to generate the SOM facies map with a 2D color map. The **crossplot** module can be found under **Display Tools** in the **aaspi_util** GUI:



The crossplot GUI is shown on the following page:



SOM axes 1 and 2 are taken as inputs for x and y axes in the crossplot. To ensure a smooth color transition, 4096 colors are used for the 2D colorbar to be generated (64 by 64 colors). The result is shown below (from a different survey in Canterbury basin, New Zealand):

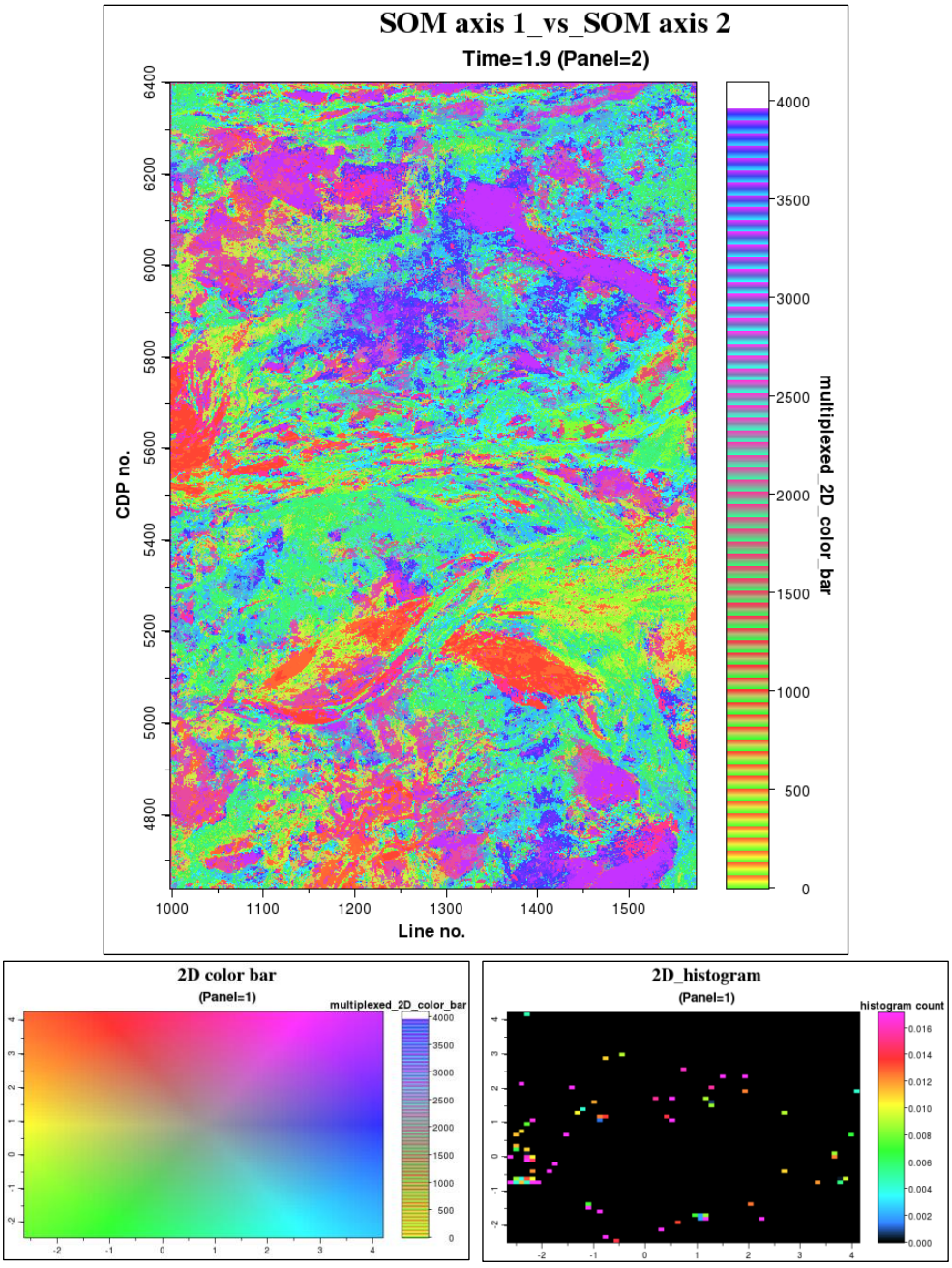


Figure 3.

From crossplotting the two SOM axes, we generate a crossplotted volume, a 2D color map, and a 2D histogram of the cross-plotted volume (Figure 3). The 2D histogram shows clusters of facies, and these clusters are color-coded by the color at the corresponding position in the 2D color map. In this example, we observe that the channels at cdp range 5000 to 5400 are in different facies, and are well separated from the surrounding sediments. Another output directly from the **som3d** module is the end member index volume as shown below (see next page):

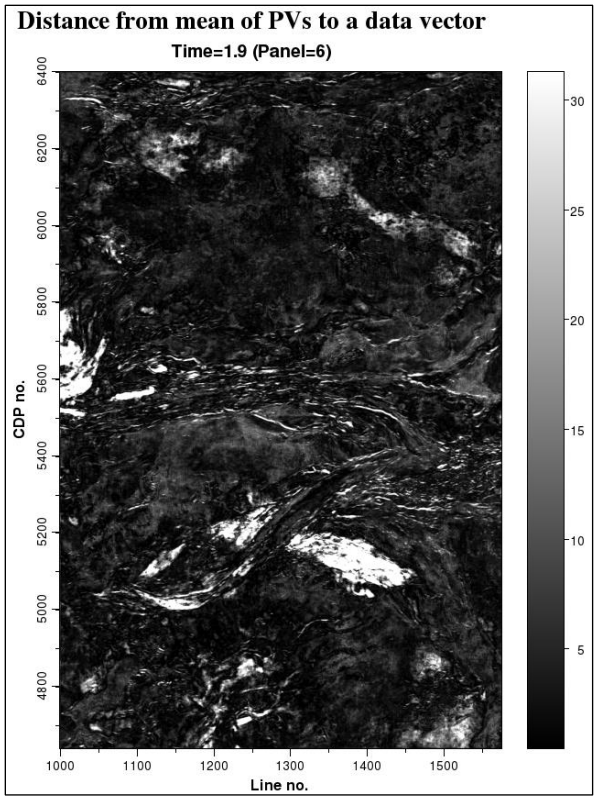


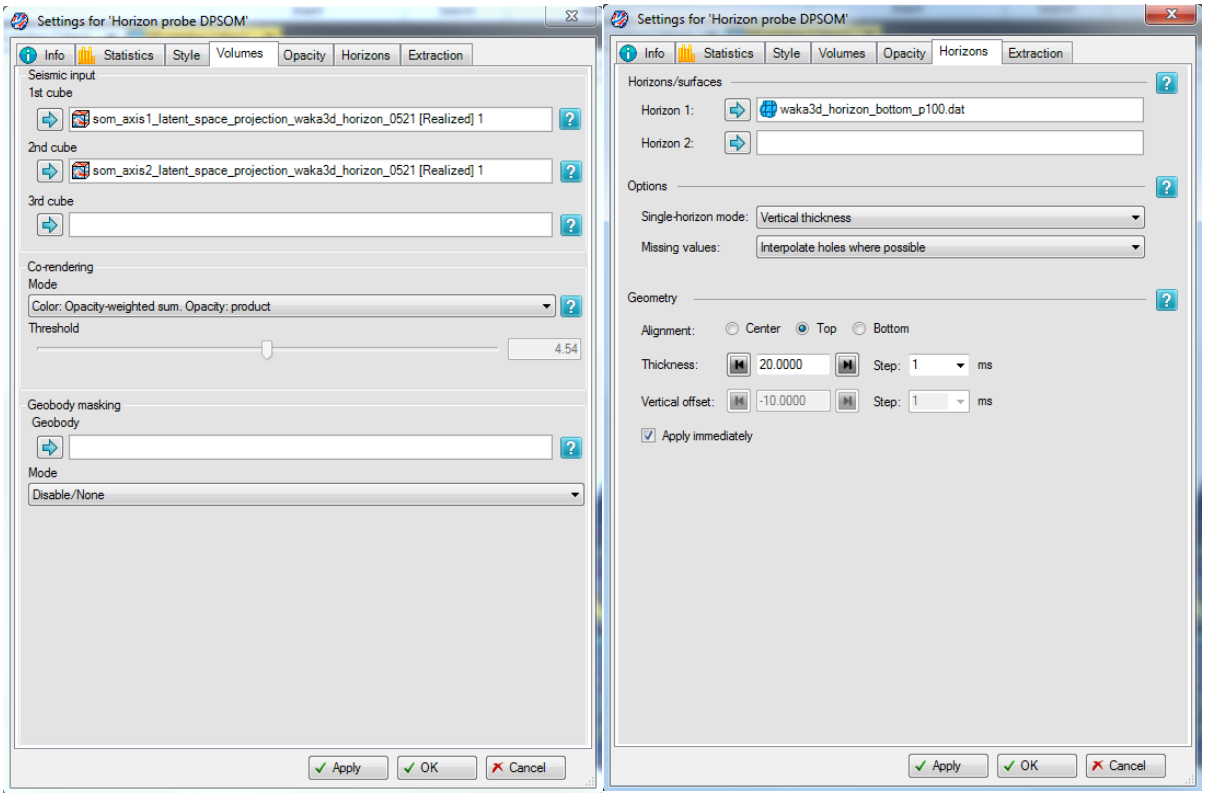
Figure 4.

This image shows the Mahalanobis distance from a multiattribute data vector to the center of all prototype vectors, providing an estimate of the likelihood of a data vector being an end member (uncommon facies). In this example, the red-colored facies has a large distance value, therefore it is considered to be a rare facies.

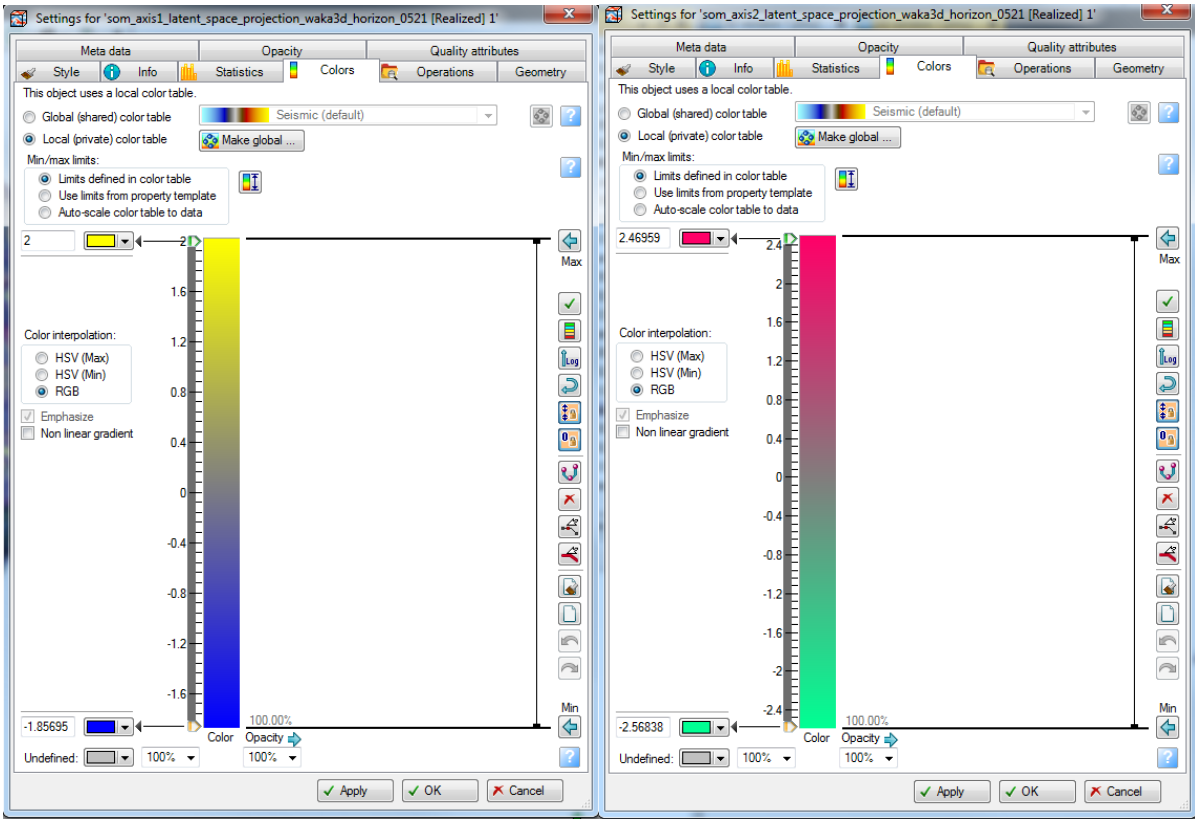
Visualization by crossplotting two SOM axes in Petrel

In the previous section, we described how to use AASPI crossplot module to generate a SOM facies volume. However, most commercial interpretation software can only display a finite number of colors if using a discrete colorbar, therefore interpreters may not be able to import and properly display the AASPI generated crossplot volume into a commercial interpretation package. To overcome this issue, instead of generating a SOM facies volume with its corresponding colorbar in AASPI, we can crossplot (corender) the two SOM axes directly in an interpretation package. In this way, we are able to generate a facies volume with much more smooth transition in color, and we also use Petrel as an example to show this trick.

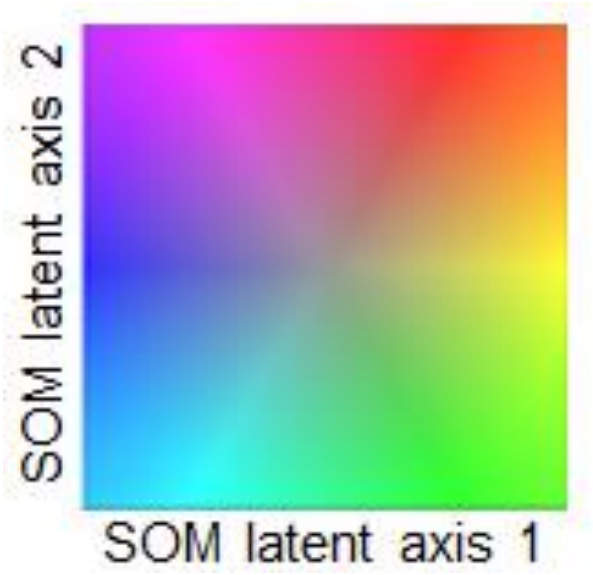
We import the two SOM axes into Petrel, and use the volumetric corendering probes to corender these two volumes. In this case, we use horizon probe as we want to display the facies along a horizon of interest. As shown in the screenshots below, we select the two SOM axes as the input volumes, and make the horizon probe aligned along the top of the horizon (in order to actually see facies on that horizon).



In order to fake a 2D colorbar, we then need to change the colorbars of the two SOM axis volumes as follows, and make the max and min value of the colorbar to best fit the data range:



In this way, we are actually faking a 2D color map like this:



And if we see the horizon probe from top (display in a 2D window in Petrel), the cross-plotting (co-rendering) result will look like the following image (see next page):

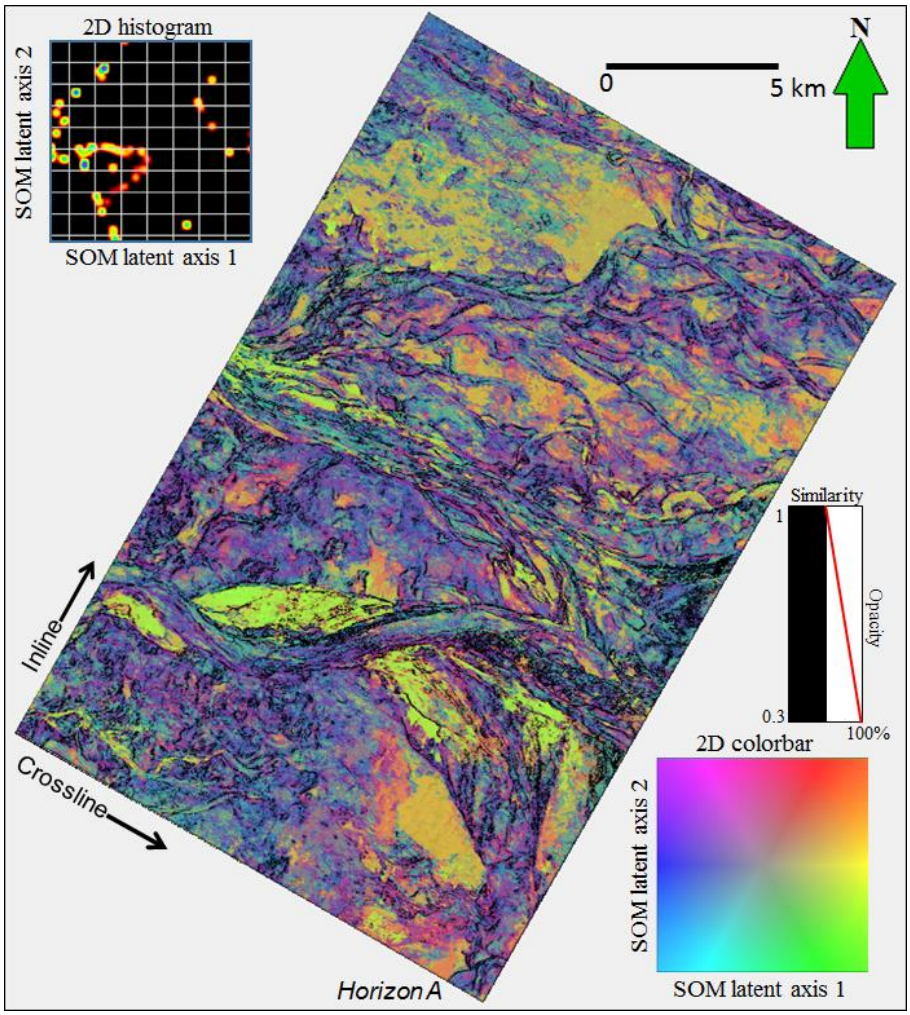


Figure 5.

In this image we are also co-rendering Sobel filter similarity to highlight the edges. To do so we extracted the Sobel filter similarity along the horizon that we used in the horizon probe, and display in the same window with the horizon probe, using an opacity curve shown in the figure. The case study in which we generated the facies map above can be found in Zhao et al. (2016).

Geobodies extraction on the facies volume in Petrel (old)

The following is a simple workflow to show the geobodies extraction in Petrel. The example is taken from a deep water Gulf of Mexico dataset (Roy et al, 2011, GCSSEPM 2011 talk). Figure 6 shows the horizon probe extracted around one of the horizons of interest. We apply transparency to the colorbar to highlight the continuous high amplitude seismic facies, which are interpreted as basin floor deposits in the survey (the blue colored seismic facies in Figure 7). Figure 8 shows the output after running automatic geobody extraction in Petrel. These geobodies gives a more quantitative estimation of the seismic facies.

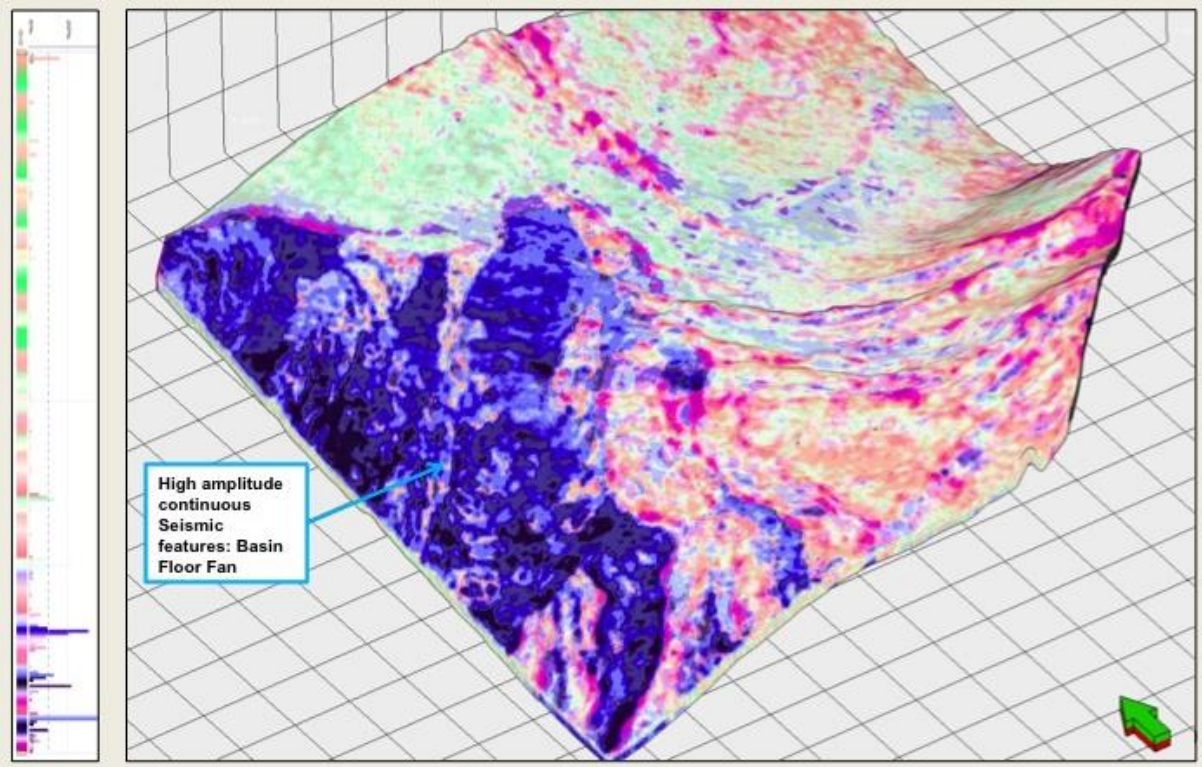


Figure 6.

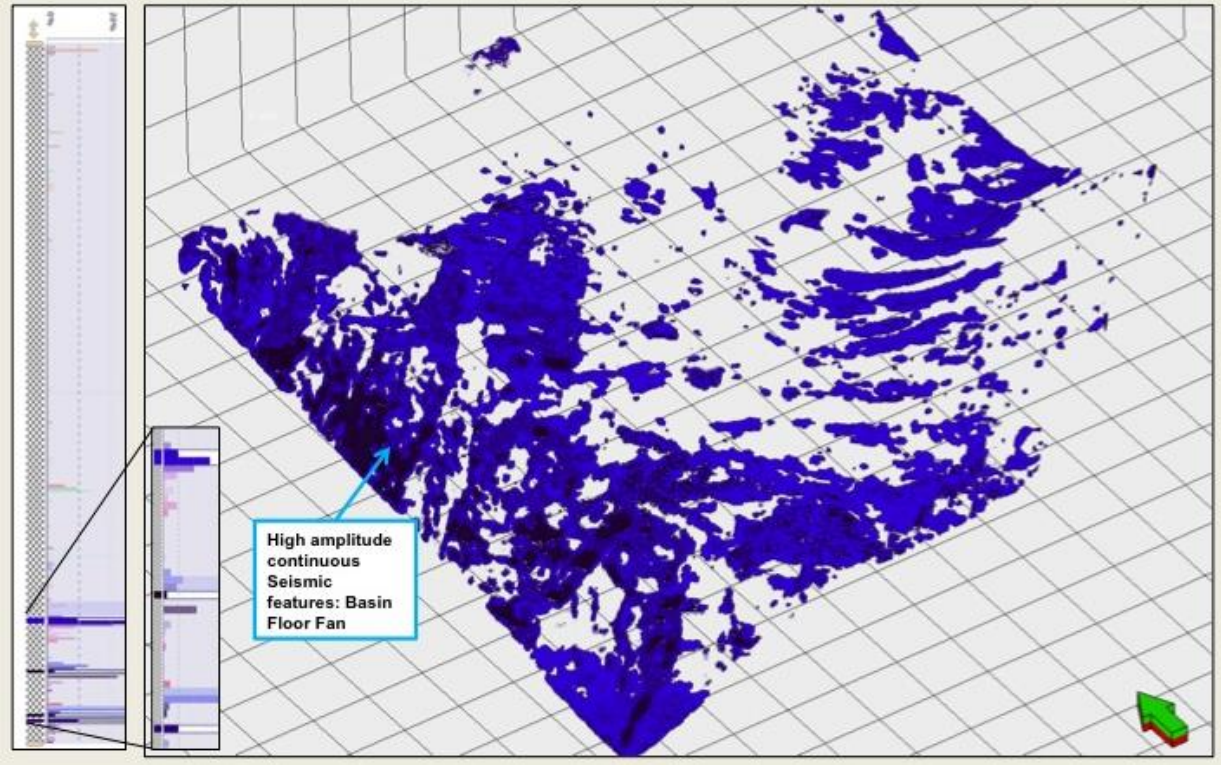


Figure 7.

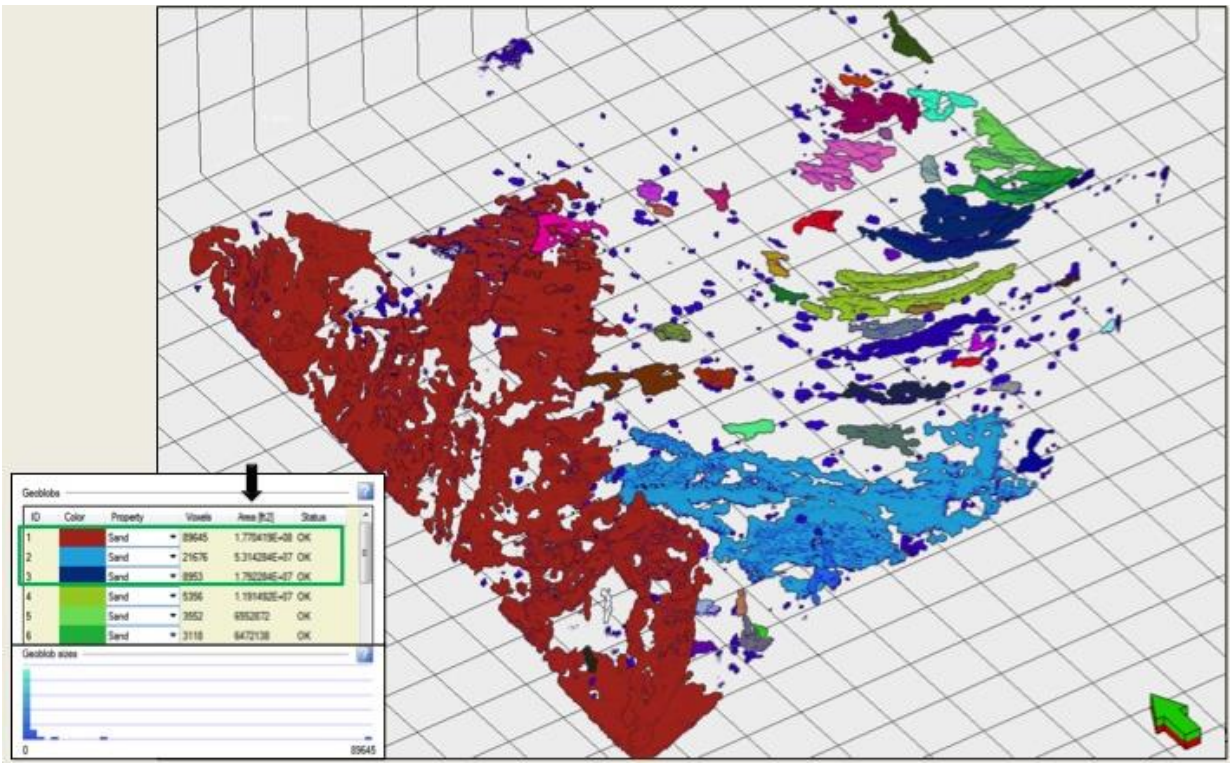
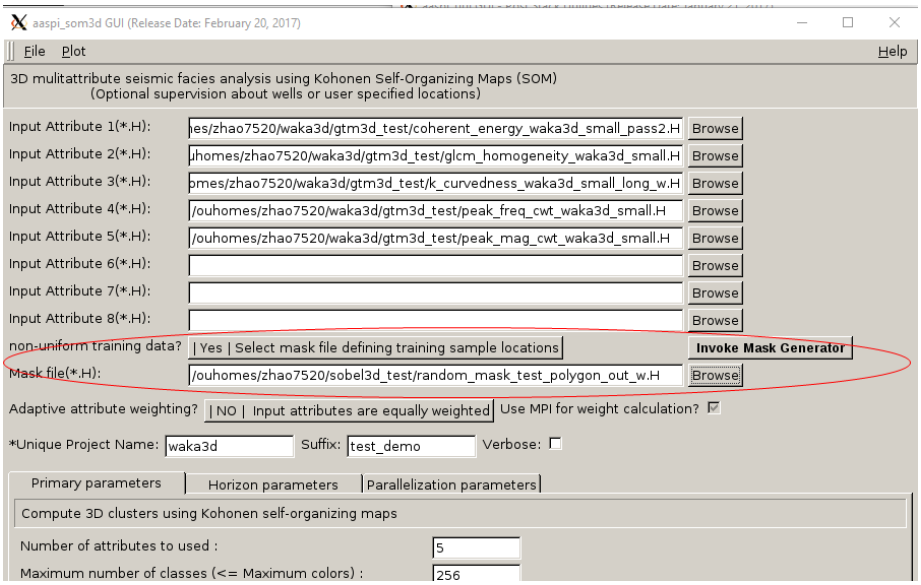


Figure 8.

Exploring advanced options

Using a mask file to extract training samples

As previously introduced, users have the option to use an externally generated mask file to define training sample locations. The following example shows how different training sample extraction leads to different SOM facies map. A mask file named as *random_mask_xxx.H* is loaded as (see next page):



For the Canterbury Basin seismic data, below is a vertical section from a mask file containing only user picked channel regions (as polygons):

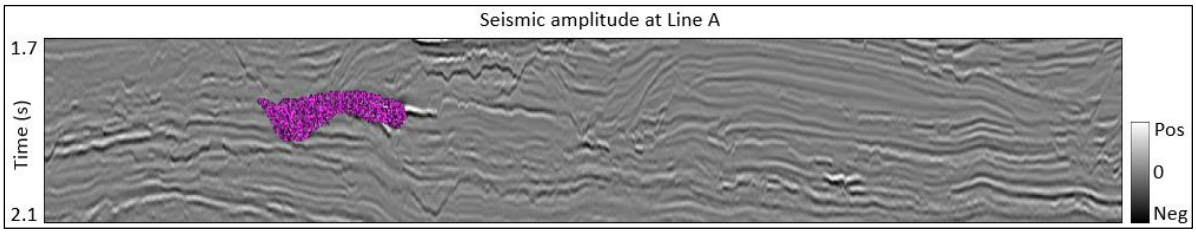


Figure 9.

The magenta dots are training samples to be used in SOM. In this particular example, we opt to randomly select 50 percent of the points within a picked (2D) polygon as training samples. Alternatively, if an interpreter prefers to include training samples representing the background, he/she can combine the picked regions with a uniformly decimated background, and assign different weights to each region and background separately:

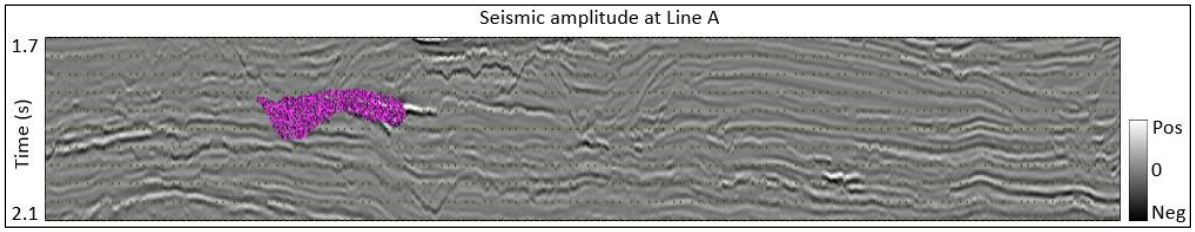


Figure 10.

In this example, magenta dots are from a user picked polygon with weight value of 2, and the regular gridded green dots form a uniform background with weight value of 1. In this way, the SOM model approximates the general trend while still emphasizes on the variations in the

channel area. We compare the SOM facies maps from these two different training sample extraction schemes along the same Horizon A which has been shown in previous figures:

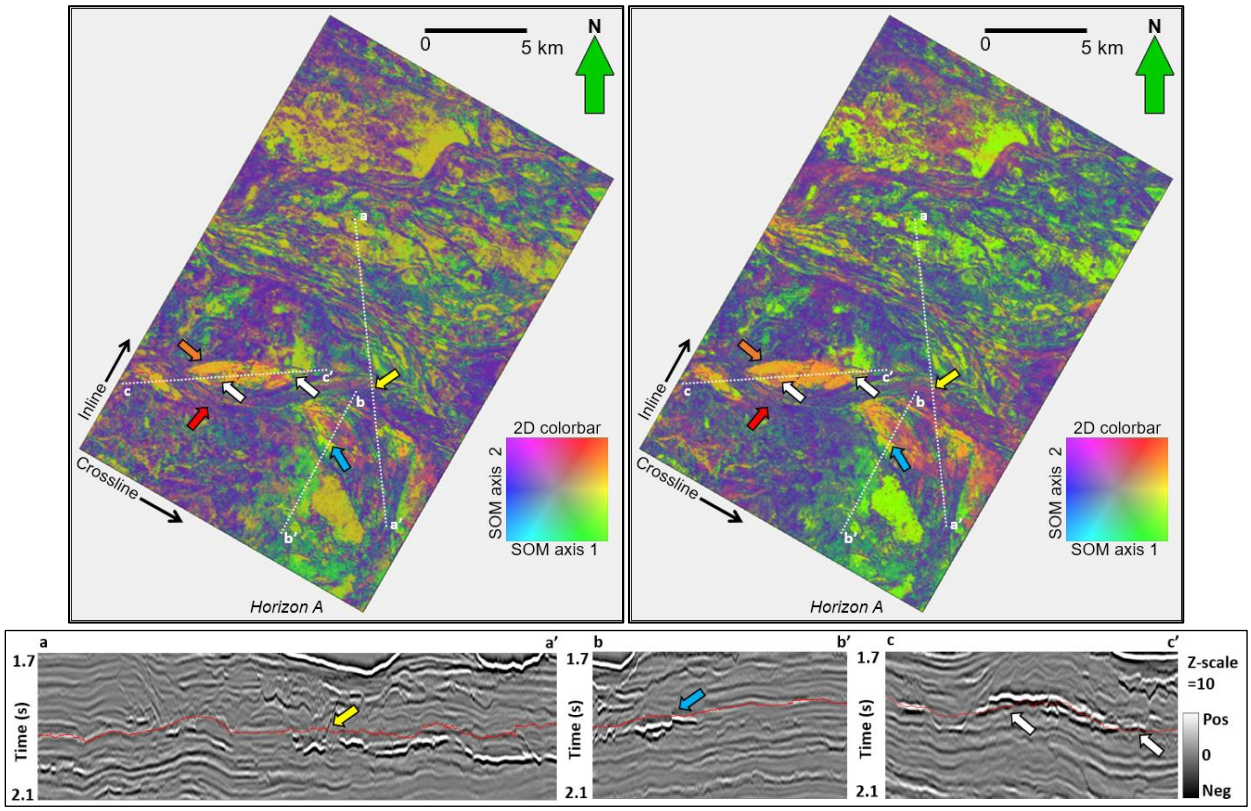


Figure 11. Left figure shows the SOM facies along Horizon A using training samples from picked polygons only. We pick training samples specifically around the two channels marked by the orange and red arrows, and by doing so, we are able to identify the subtle changes marked by the white, yellow, and blue arrows. On the other hand, because the training samples are selected around the two channels, the colors at regions outside this relatively small area do not necessary represent the real facies. Instead, they only represent the similarity between a facies and facies within the dual-channel region. The Right figure shows the SOM facies along Horizon A, using training samples from both picked polygons and the background. We take samples from uniform sampling and assign weight value of 1, and assign weight of 2 for user-picked points. We are able to observe some of the subtle changes within the dual-channel system (e.g. the edge between two channel stories marked by the blue arrow). At the same time, we still preserve the meaning of the facies outside the dual-channel region, because the majority of training samples are extracted outside the dual-channel region.

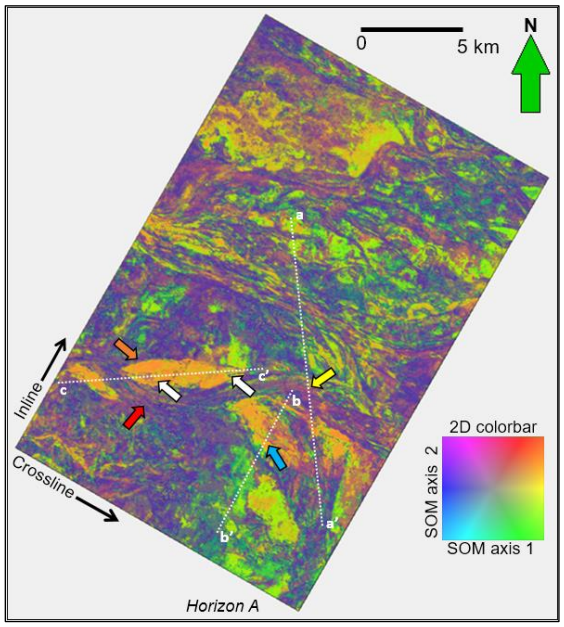
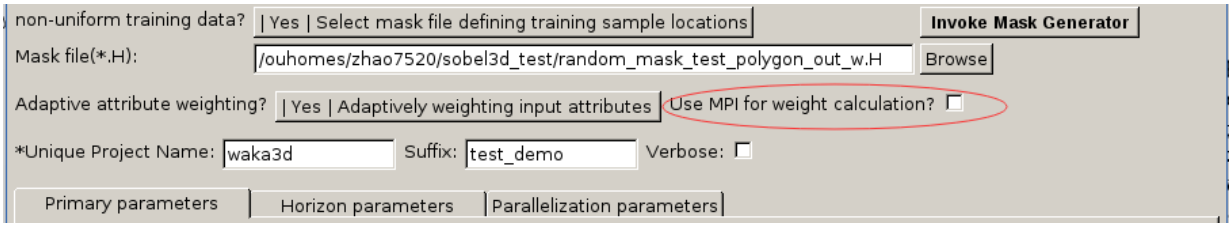


Figure 12. As a comparison, Figure 12 on the left is a SOM facies solely from uniformly sampled training data. As we expected, uniform sampling preserves the main features, such as the channel in orange (orange arrow) that is cut through by a younger channel (red arrow), and the sinuous channel complex in the north. However, there is little to no evidence for the subtle features that are marked by arrows, including a channel boundary (yellow arrow), an edge between two channel stories (blue arrow), and changes in reflection characteristic (white arrows). These features are readily visible in the previous SOM facies maps.

Adaptive attribute weighting

Note: This function is memory intensive by default. If memory related errors occur as a result of using adaptive attribute weighting, please try to uncheck the **Use MPI for weight calculation** box to run the weighting step in sequential mode. However, opting to run in sequential mode will increase the time cost considerably.



Typically, interpreters qualitatively choose input attributes for multiattribute facies analysis based on their experience and geologic target of interest. In this SOM facies analysis module, we augment this qualitative attribute selection process with quantitative measures of which candidate attributes best differentiate features of interest, by weighting input attributes based on their response from the unsupervised learning algorithm that used to generate the facies map, as well as their statistical behaviors.

We use an example from the Barnett Shale to demonstrate the effect of input attribute weighting, and compare with SOM facies map from equally weighted input attributes. In our example, the Barnett Shale lies directly on top of the dolomitic Ellenburger formation in the western region of the Fort Worth Basin. The Ellenburger formation is highly deformed, with extensive development of karst and joints that extend upwards from the water-saturated

Ellenburger into the Barnett Shale, posing drilling and completion hazards (Pollastro et al., 2007). Our objective is to use spectral decomposition, geometric, and texture attributes, which are sensitive to strata thickness, lithology, and structural deformation, to illuminate the architectural elements presented in the shallow part of Ellenburger formation. The figures below provide co-rendered attributes along a phantom horizon (Horizon A) 25 ms below the top of Ellenburger, on which we observe karst features either in isolated circular to oval shape, or in a cellular network of polygonal karst. Structural curvature defines the extension of karst regions (top left), while amplitude curvature highlights the small scale collapse (top right). We observe that highly karsted regions exhibit lower frequency compared to the surrounding area, possibly due to the non-specular scattering from the chaotic reflectors. These regions are also low in peak spectral magnitude as a substantial amount of the reflected energy is not properly received by the receivers within the migration aperture.

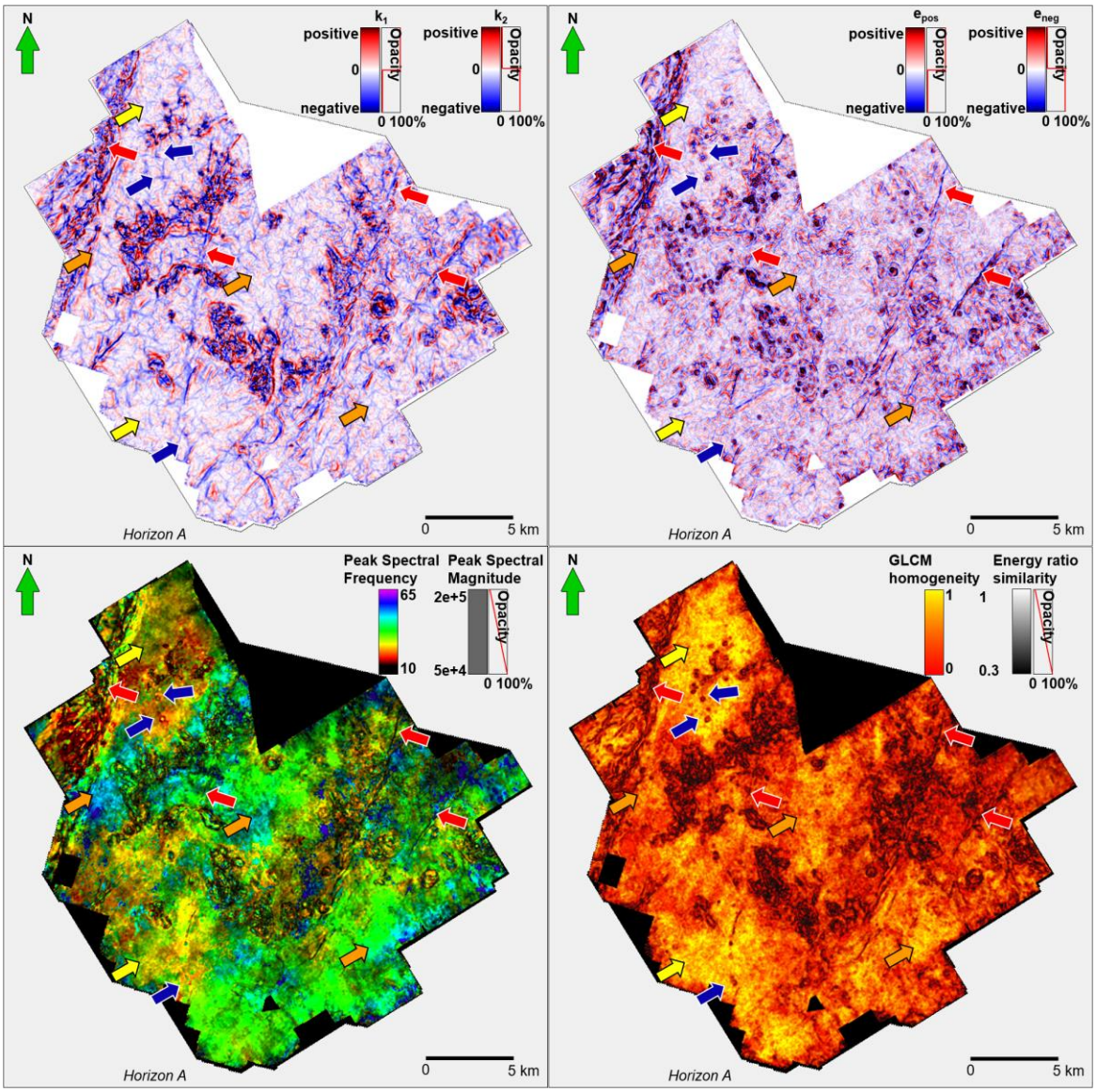


Figure 13.

When executing the program, the program will build multiple SOM models with different subgroups of input attributes to estimate the attribute weights, and the attribute weights are calculated once all attribute combinations have been tested:

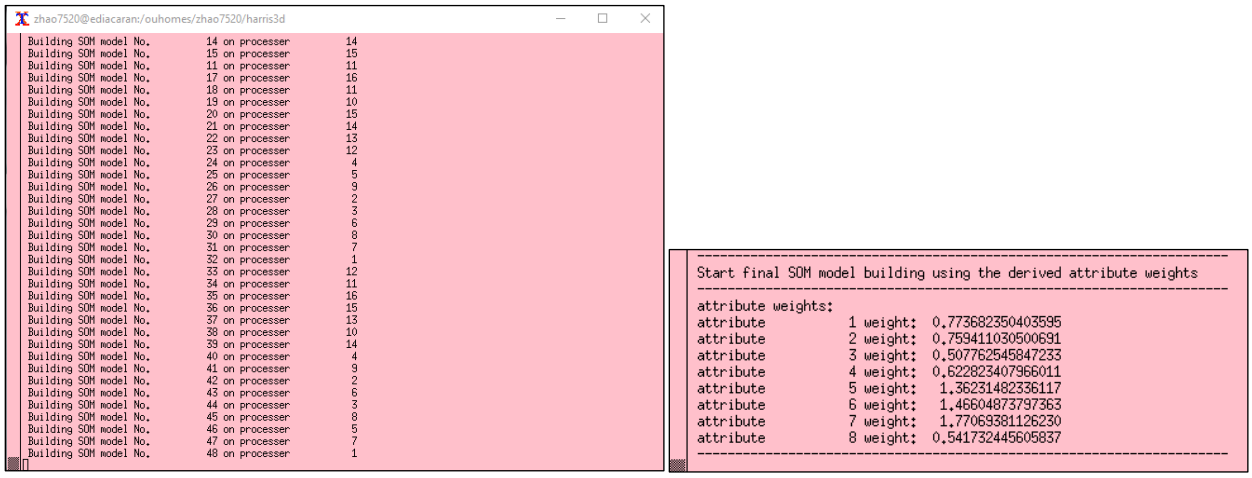


Figure 13 below are the SOM facies maps from adaptively weighted attributes (left) and (by default) equally weighted attributes (right):

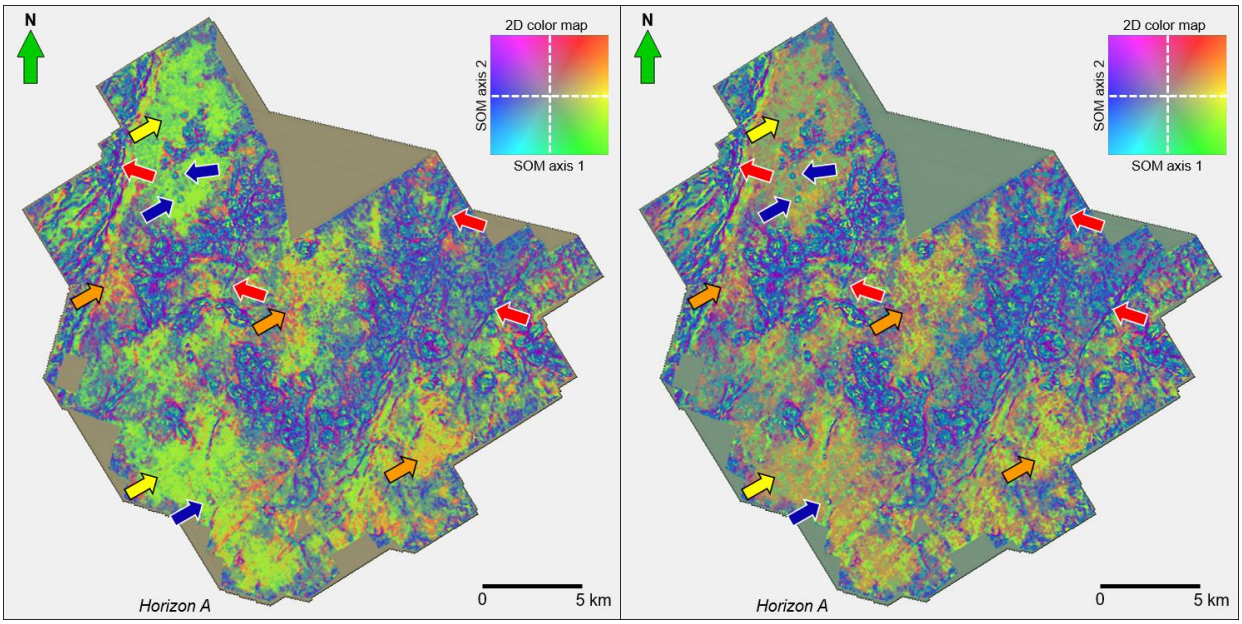


Figure 13.

Comparing the two figures above (Figure 13), we observe that both SOM facies maps are able to delineate the karst, faults, and fractures equally well. This observation verifies an assumption that adding a penalty weight does not significantly alter the curvature and similarity anomaly contributions. The polygonal karst regions are characterized by purple and cyan facies, where purple corresponds to anticlinal components and cyan synclinal components. Compared to the co-rendered structural curvatures, both SOM facies maps provide details about smaller scaled

karst caves that are not identifiable on structural curvatures, most of which correspond to fracture joints (blue arrows). We are also able to identify the major faults (red arrows) close to the polygonal karst regions, suggesting a tectonic control of the karst development (Qi at al., 2014). The main difference between adaptively weighted and equally weighted SOM comes from regions marked with yellow and orange arrows. In the left figure, the yellow arrow regions are in a lime green facies, where the orange arrow regions are in an orange facies. In contrast, these regions look nearly identical in the right figure, all being brownish cellular textures that somehow follow the trend on the curvature attributes. The lime green versus orange facies change in the left figure reflects the frequency variation found in the previous peak frequency attribute, where low peak frequency regions are in lime green facies (yellow arrows), and high frequency regions are in orange facies (orange arrows). The peak frequency provides information on tuning thickness, which adds another dimension besides surface morphology. The SOM facies map from equally weighted attributes, on the other hand, does not distinct such frequency variation clearly.

References

Coleou, T., M. Poupon, and K. Azbel, 2003, Unsupervised seismic facies classification: A review and comparison of techniques and implementation: The Leading Edge, **22**, 942-953.

Gao, D., 2007, Application of three-dimensional seismic texture analysis with special reference to deep-marine facies discrimination and interpretation: An example from offshore Angola, West Africa: AAPG Bulletin, **91**, 1665-1683.

Kohonen, T., 1982 Self-organized formation of topologically correct feature maps: Biological Cybernetics, **43**, 59-69.

Kohonen,T., 2001, Self-organizing Maps, 3rd ed.: Springer-Verlag.

Matos, M. C., K. J. Marfurt., and P. R. S. Johann, 2009, Seismic color Self-Organizing Maps: 11th International Congress of the Brazilian Geophysical Society, Expanded Abstracts.

Matos, M. C., P. L. Osorio, and P. Johann, 2007, Unsupervised seismic facies analysis using wavelet transform and self-organizing maps: Geophysics, **72**, P9-P21.

Qi, J., B. Zhang, H. Zhou, and K. Marfurt, 2014, Attribute expression of fault-controlled karst — Fort Worth Basin, Texas: A tutorial: Interpretation, **2**, SF91-SF110.

Roy, A., and K. J. Marfurt, 2010, Applying self-organizing maps of multiattributes, an example from the Red-Fork Formation, Anadarko Basin: 81st Annual International Meeting Society of Exploration Geophysicists, Expanded Abstracts, 1591-1595.

Roy, A., K. J. Marfurt, and M. C. Matos, Application of 3D clustering analysis for deep water marine Seismic facies classification-an example from deep water Northern Gulf of Mexico, GCSSEPM 2011.

Strecker, U., and R. Uden, 2002, Data mining of 3D poststack attribute volumes using Kohonen self-organizing maps: The Leading Edge, **21**, 1032-1037.

Wallet, C. B., M. C. Matos, and J. T. Kwiatkowski, 2009, Latent space modeling of seismic data: An overview: The Leading Edge, **28**, 1454-1459.

Zhao, T., J. Zhang, F. li, and K. J. Marfurt, 2016, Characterizing a turbidite system in Canterbury basin, New Zealand using seismic attributes and distance-preserving self-organizing maps: Interpretation (accepted, will appear in February 2016 issue).