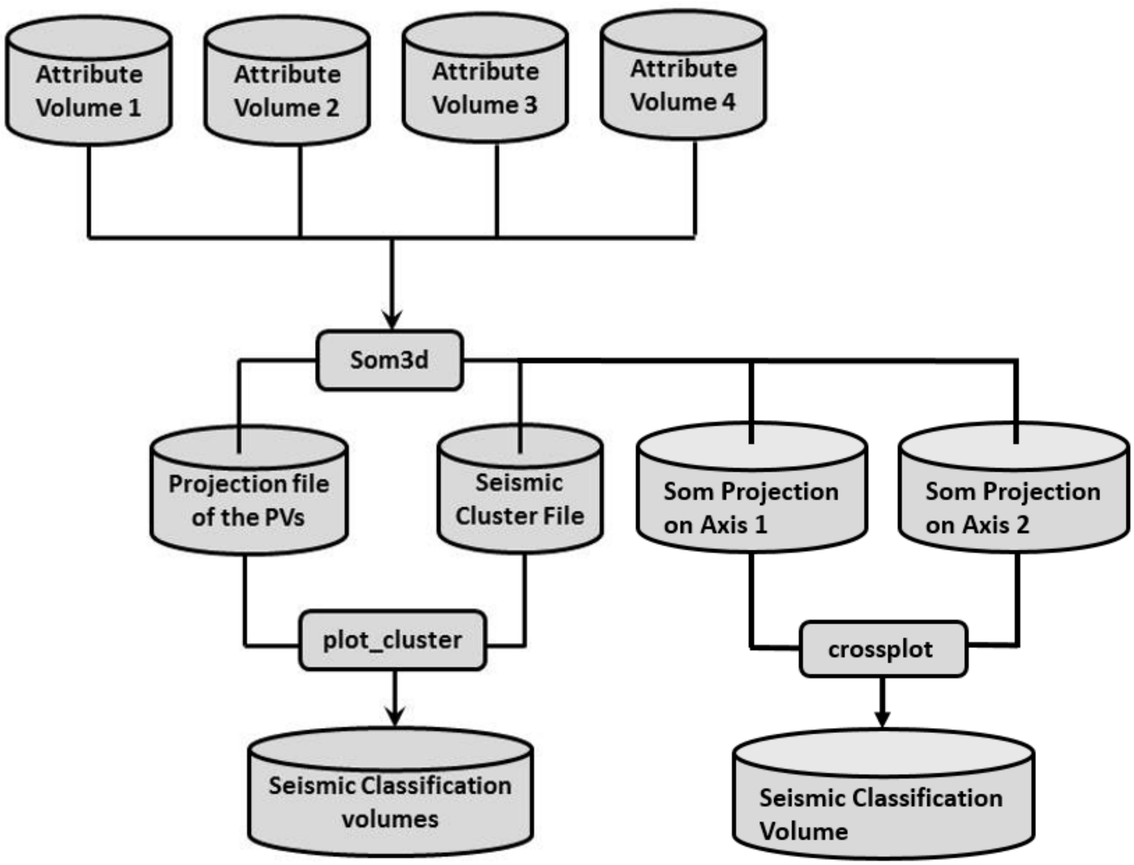
Volumetric Self-Organizing Maps for 3D SEISMIC FACIES ANALYSIS – PROGRAM som3D

Computation flow chart

This 3D Facies classification analysis is comprised of two separate modules; **som3d** and **plot_clusters**. The **multiattribute_pca** is now internal to the **som3d** program, which preconditions the different input attribute volumes that serve as input to **som3d**. It calculates the eigenvalues and eigenvectors from input dataset, which will be used to project the input data vector into the latent space. The last module - **plot_cluster** assigns colors to the different trained facies into a 2D RGB gradational scale and plots the output seismic facies volume (Matos et al., 2009, Roy et al., 2011). The **som3d** program will also 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. If using such crossplot for visualization, **plot_clusters** is not necessary to use. Below is the flowchart showing the workflow of 3D seismic facies analysis.



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 $j=1,2,\ldots,J$ 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 $i=1, 2, \ldots, P$ 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, $\mathbf{m}_b$:

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

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

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

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

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

$$h_{bi}(t) = e^{-(\|\mathbf{r_b} - \mathbf{r_i}\|^2 / 2\sigma^2(t)} \quad \text{, and} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

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

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 with for 2D spaces will be defined as hue, $\mathcal{H}$,

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

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

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

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):

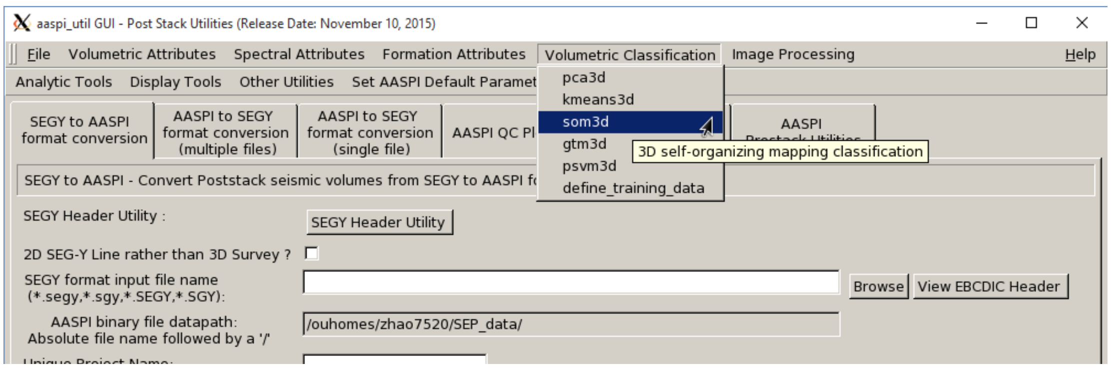
$$\mathbf{p}_k(t+1) = \mathbf{p}_k(t) + \alpha(t)\cdot\left(1-\frac{\delta_{vk}}{d_{vk}}\right)\cdot\left(\mathbf{p}_v(t)-\mathbf{p}_k(t)\right),\ \forall\, k \neq v. \qquad \text{…………………………...(7)}$$

In Equation 7, $\mathbf{p}_k(t)$ is the positon of an SOM node before adjustment; $\mathbf{p}_k(t+1)$ is the position of an SOM node after adjustment; $\mathbf{p}_v(t)$ is the position of the current winning node; δ_{vk} and d_{vk} are the distance between an SOM node and the current winning node in input data space and 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) location in our volume (which are actually the normalized input attribute values at that location). We call this new volume the normalized multi-attribute volume and project it onto a 2D latent space by 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

$$\|\mathbf{x} - \mathbf{m}_b'\| = \min\{\|\mathbf{x} - \mathbf{m}_i'\|\} \quad \text{...} (7)$$

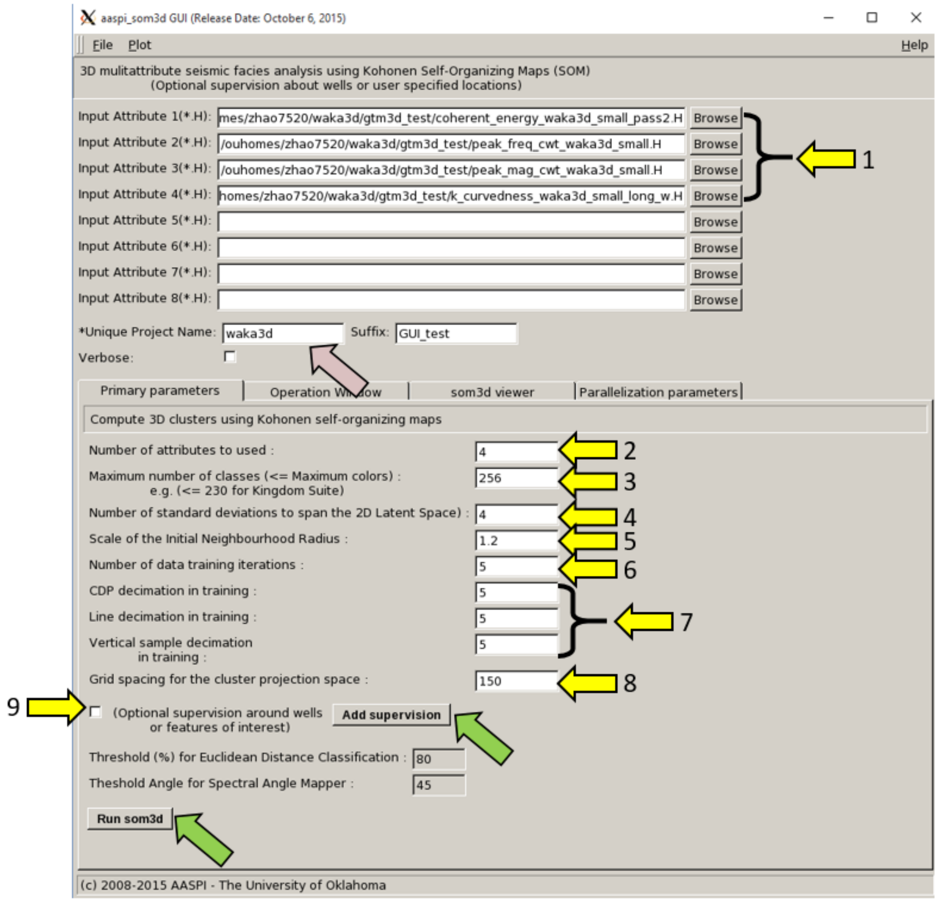
where $\mathbf{m}_b{}'$ is the nearest PV to the input data sample vector $\mathbf{x}$. Each voxel is then assigned the color of $\mathbf{m}_b'$. In this manner, two dissimilar neighboring samples in the seismic volume will be far apart in the latent space and have different colors. Conversely, two similar 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.



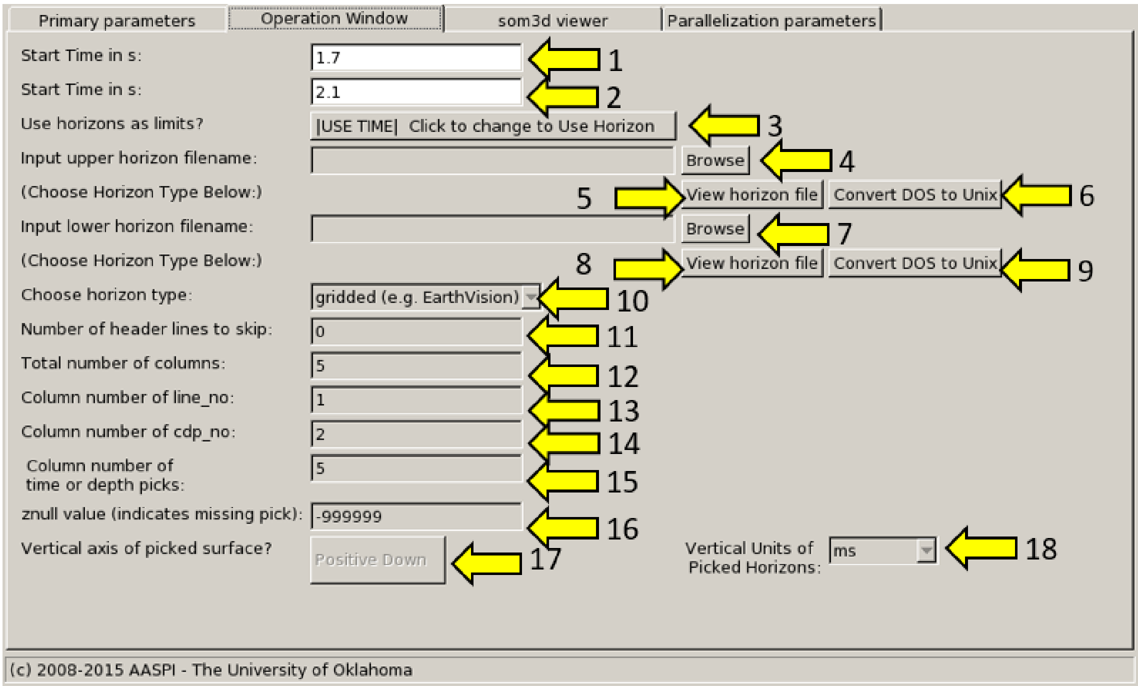
This Program **som3d** is launched from the *3D facies classification* in the main **aaspi_util** GUI

Computing som3d module

Setting the primary parameters is the first step of 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. Thus in this case we take 4096 classes, which later can be represented by a 64 – *by* – 64 colorscale. 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.


Primary parameters
Operation Window
som3d viewer
Parallelization parameters
Start Time in s:
1.7
1
Start Time in s:
2.1
2
Use horizons as limits?
|USE TIME| Click to change to Use Horizon
3
Input upper horizon filename:
Browse
4
(Choose Horizon Type Below:)
5
View horizon file
Convert DOS to Unix
6
Input lower horizon filename:
Browse
7
(Choose Horizon Type Below:)
8
View horizon file
Convert DOS to Unix
9
Choose horizon type:
gridded (e.g. EarthVision)
10
Number of header lines to skip:
0
11
Total number of columns:
5
12
Column number of line_no:
1
13
Column number of cdp_no:
2
14
Column number of time or depth picks:
5
15
znull value (indicates missing pick):
-999999
16
Vertical axis of picked surface?
Positive Down
17
Vertical Units of Picked Horizons:
ms
18
(c) 2008-2015 AASPI - The University of Oklahoma

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 have are defined by *line_no, cdp_no (crossline_no)* and *time* triplets for each trace location. Examples of both format 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*).

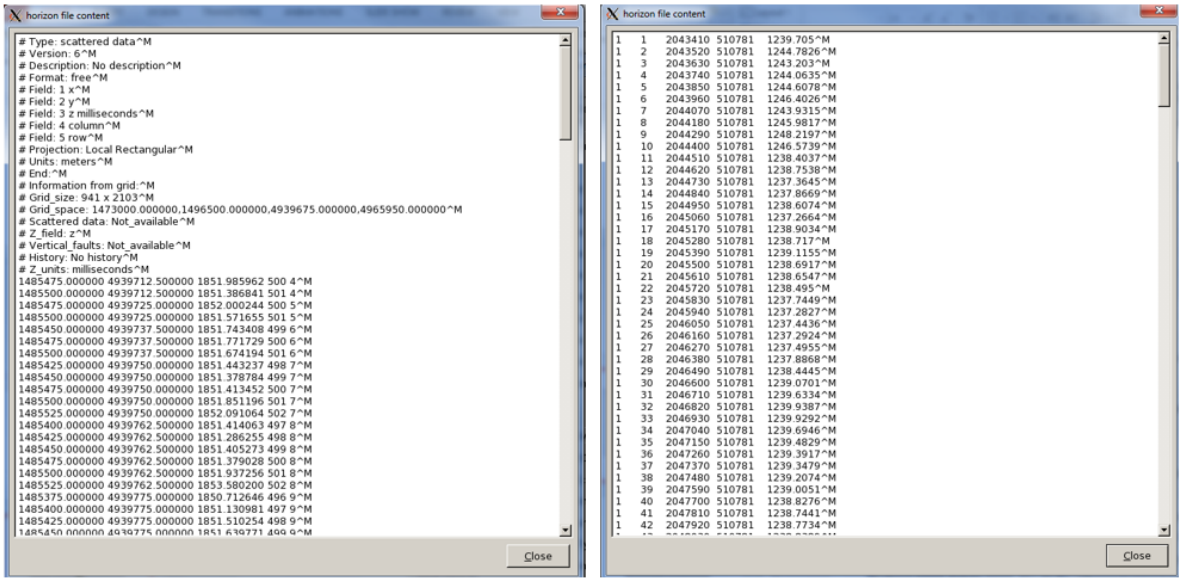
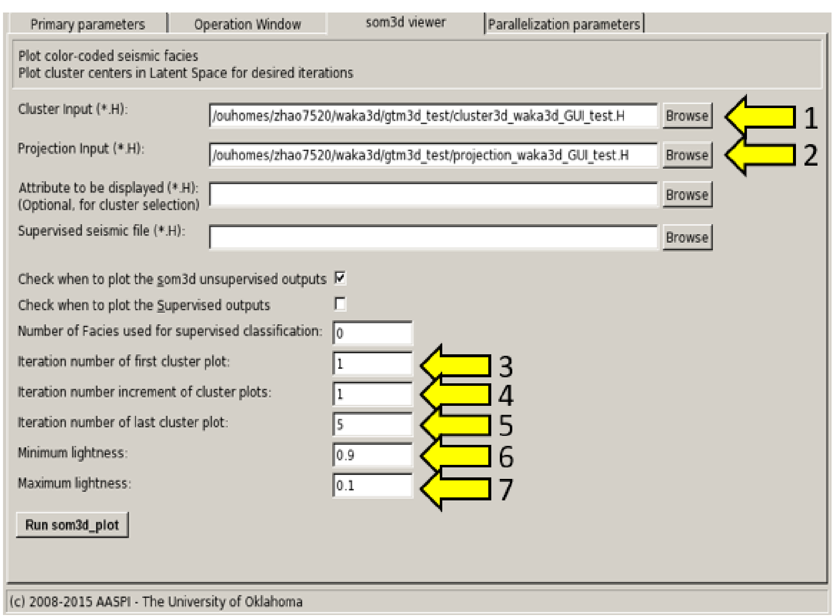


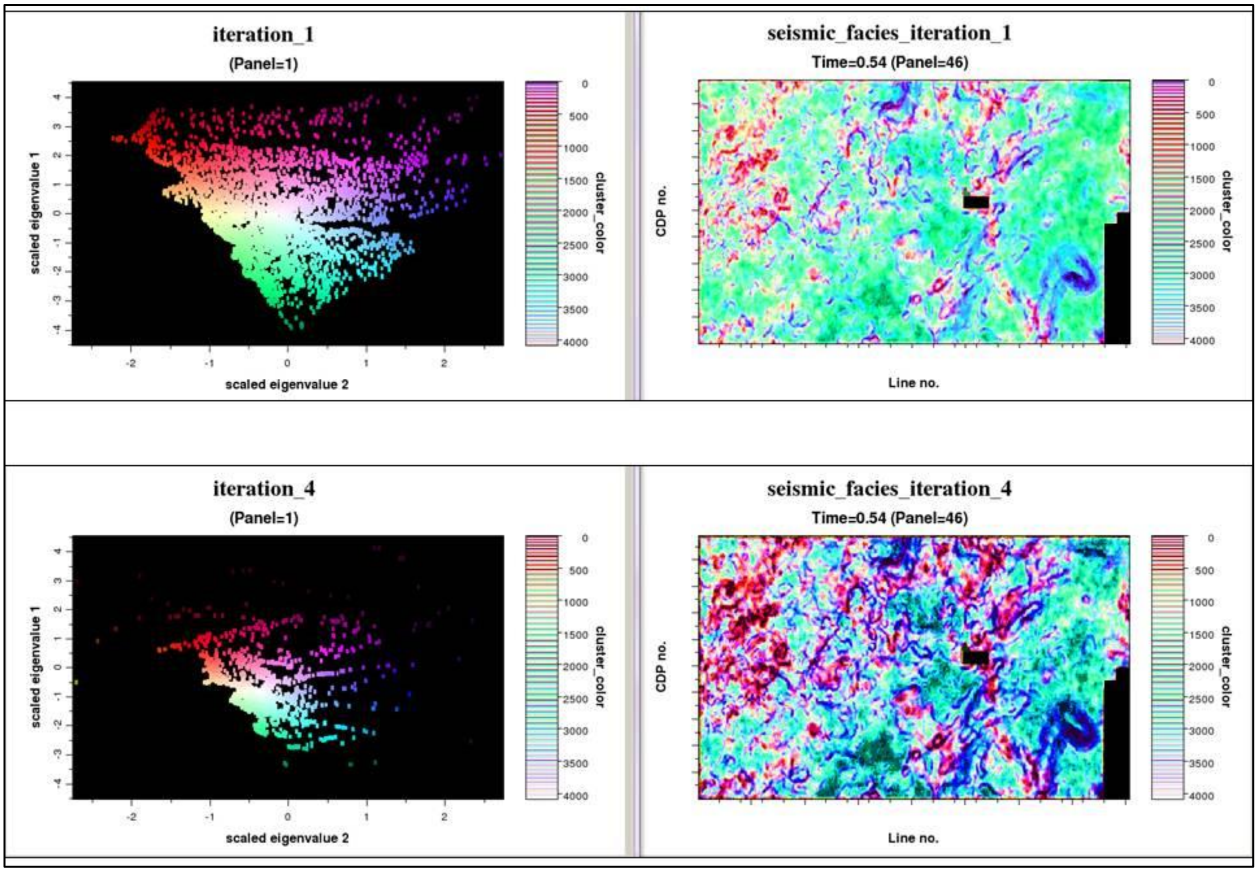
Figure 1. (left) A gridded horizon file (EarthVision format). (right) An interpolated horizon file with five columns (ASCII free format).

Plotting (1): plot_clusters module (old)

The 2nd step is to color the trained 3D seismic dataset and highlight the variation in seismic facies. Different facies are represented by different colors. The *SOM3D Viewer* is the next tab. The outputs from SOM3D serve as input to this module (*Arrow 1 and 2*). This module helps QC the facies volume after each training. This also generates a suite of color files, which can be taken as input in visualization software like Petrel.



Step 2(1): This step creates the various color-files and colors the projected trained vectors and the 3D trained dataset. The cluster file (*cluster3d**) and the projection file (*project3d**) are taken as input. The Minimum lightness and the maximum lightness values help in changing the minimum and maximum saturation value of the 2D gradational colorbar. The *.alut file generated by this module can be imported into Petrel (shown in the later section) for 3D visualization.

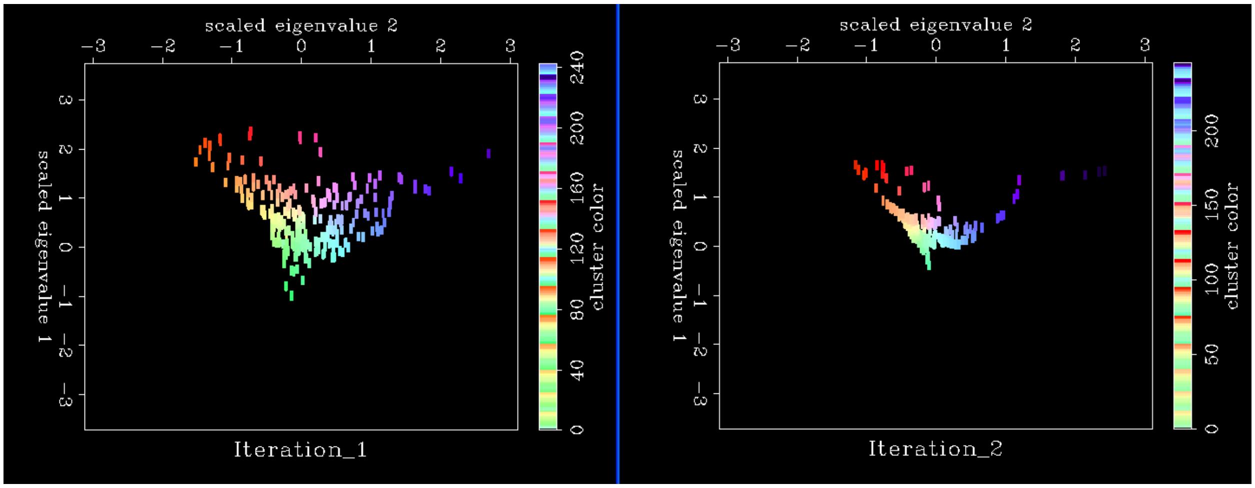


Above is the output generated from the SOM3D facies clustering module with 4096 initial classes. Note that clusters start forming for iteration 4 and the overdefined 4096 classes are actually represented by 4-5 colors. The volumes used in this analysis are the dip magnitude, sobel edge detecting filter, GLCM entropy and GLCM dissimilarity. Thus this selection will represent both structural and depositional features together in the seismic facies volume. Note that the dark bule component in the seismic facies volumes are the stuctural features representing high dip or faults or fractures. The depositional facies are in light or dark green and red. This seismic facies represents a timeslice within the Mississippian Lime. Thus this seismic facies volume when tied to well information can represent the rock types in the survey. The input volumes should be selected knowing the requirements of the outputs. For example the facies volume generated by using the inversion, lambda-rho and mu-rho volumes will represent more variation in the rock types based on the porosity and clay content.

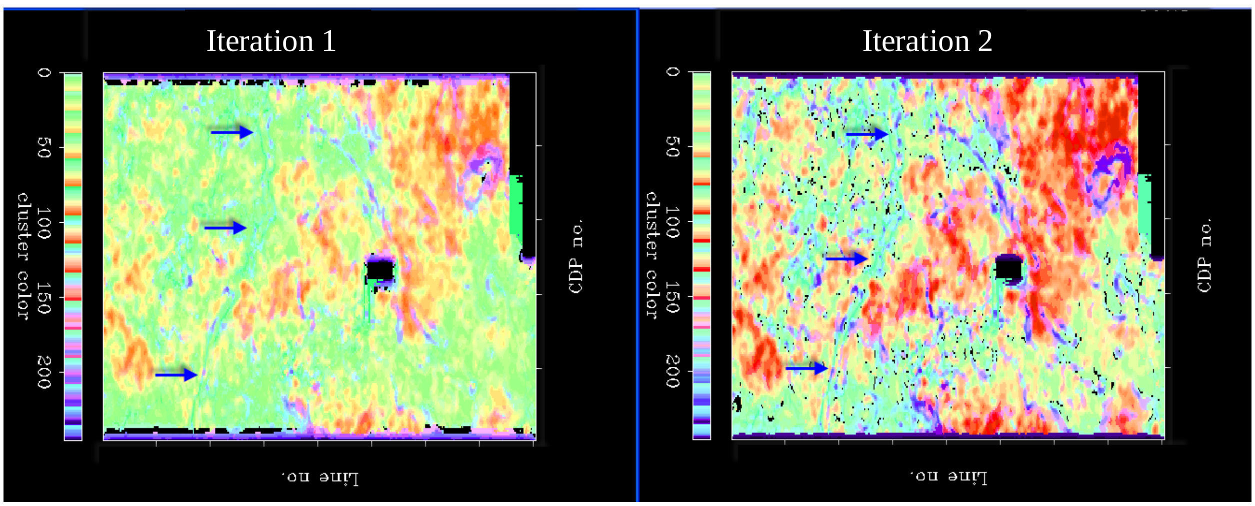
Executing this module will generate the following outputs as shown below. The following dataset from a Mississippian lime is shown as an example. Below are the list of files that are created at the end. The *som_waveform** is the final clustered file which should be made .segy for interpretating the results in any vizualization software, the *som_project*ascii* file is the initial set of projections with the untrained prototype vectors. It forms a elliptical grid of projection points. The *project3d** and *cluster3d** are the clusterd SEP files after each iteration and are used to create separate colobars after each iteration with the *plot_cluster* program. The *p_vector** are the ascii files with the projection of the trained clusters prototype vectors. The plot_cluster program generates the **sep* and the **alut* and other colorbars. Different colorbar sets are created for each iterartions.

```
roy5699@opal:~/projects/personia/personia_mc/som3d_6
-rw-rw-r--  1 roy5699 roy5699 2.3K Oct  2 12:03 multti_attr_eigenvector_MC_6.H@@
-rw-rw-r--  1 roy5699 roy5699 4.1K Oct  2 12:03 multti_attr_eigenvector_MC_6.H
-rw-rw-r--  1 roy5699 roy5699 2.3K Oct  2 12:03 multti_attr_eigenvalue_MC_6.H@@
-rw-rw-r--  1 roy5699 roy5699 4.1K Oct  2 12:03 multti_attr_eigenvalue_MC_6.H
-rw-rw-r--  1 roy5699 roy5699  13K Oct  2 12:05 multiattribute out
-rw-rw-r--  1 roy5699 roy5699  450 Oct  2 12:05 som_waveform_MC_6.H
-rw-rw-r--  1 roy5699 roy5699 2.3K Oct  2 12:05 pv_MC_6__5.H@@
-rw-rw-r--  1 roy5699 roy5699 4.1K Oct  2 12:05 pv_MC_6__5.H
-rw-rw-r--  1 roy5699 roy5699 2.3K Oct  2 12:05 pv_MC_6__4.H@@
-rw-rw-r--  1 roy5699 roy5699 4.1K Oct  2 12:05 pv_MC_6__4.H
-rw-rw-r--  1 roy5699 roy5699 2.3K Oct  2 12:05 pv_MC_6__3.H@@
-rw-rw-r--  1 roy5699 roy5699 4.1K Oct  2 12:05 pv_MC_6__3.H
-rw-rw-r--  1 roy5699 roy5699 2.3K Oct  2 12:05 pv_MC_6__2.H@@
-rw-rw-r--  1 roy5699 roy5699 4.1K Oct  2 12:05 pv_MC_6__2.H
-rw-rw-r--  1 roy5699 roy5699 2.3K Oct  2 12:05 pv_MC_6__1.H@@
-rw-rw-r--  1 roy5699 roy5699 4.1K Oct  2 12:05 pv_MC_6__1.H
-rw-rw-r--  1 roy5699 roy5699 2.3K Oct  2 12:05 pc_MC_6__5.H@@
-rw-rw-r--  1 roy5699 roy5699 4.1K Oct  2 12:05 pc_MC_6__5.H
-rw-rw-r--  1 roy5699 roy5699 2.3K Oct  2 12:05 pc_MC_6__4.H@@
-rw-rw-r--  1 roy5699 roy5699 4.1K Oct  2 12:05 pc_MC_6__4.H
-rw-rw-r--  1 roy5699 roy5699 2.3K Oct  2 12:05 pc_MC_6__3.H@@
-rw-rw-r--  1 roy5699 roy5699 4.1K Oct  2 12:05 pc_MC_6__3.H
-rw-rw-r--  1 roy5699 roy5699 2.3K Oct  2 12:05 pc_MC_6__2.H@@
-rw-rw-r--  1 roy5699 roy5699 4.1K Oct  2 12:05 pc_MC_6__2.H
-rw-rw-r--  1 roy5699 roy5699 2.3K Oct  2 12:05 pc_MC_6__1.H@@
-rw-rw-r--  1 roy5699 roy5699 4.1K Oct  2 12:05 pc MC 6  1.H
-rw-rw-r--  1 roy5699 roy5699  11K Oct  2 12:06 som project MC 6.ascii
-rw-rw-r--  1 roy5699 roy5699  413 Oct  2 12:07 projec3d_MC_6.H
-rw-rw-r--  1 roy5699 roy5699  431 Oct  2 12:07 cluster3d_MC_6.H
-rw-rw-r--  1 roy5699 roy5699  153 Oct  2 12:07 [illegible]
-rw-rw-r--  1 roy5699 roy5699  11K Oct  2 12:07 p_vector_1
-rw-rw-r--  1 roy5699 roy5699  11K Oct  2 13:00 p vector 2
-rw-rw-r--  1 roy5699 roy5699 272K Oct  2 13:53 som3d_out
-rw-rw-r--  1 roy5699 roy5699  35K Oct  2 14:02 som3d_plot_out
-rw-rw-r--  1 roy5699 roy5699 7.8K Oct  2 14:02 som3d colors 6 2.voxelgeo
-rw-rw-r--  1 roy5699 roy5699 7.3K Oct  2 14:02 som3d_colors_6_2.sep
-rw-rw-r--  1 roy5699 roy5699 4.0K Oct  2 14:02 som3d_colors_6_2.pal
-rw-rw-r--  1 roy5699 roy5699 9.4K Oct  2 14:02 som3d_colors_6_2.landmark
-rw-rw-r--  1 roy5699 roy5699 7.3K Oct  2 14:02 som3d_colors_6_2.iesx
-rw-rw-r--  1 roy5699 roy5699  11K Oct  2 14:02 som3d_colors_6_2.gpc
-rw-rw-r--  1 roy5699 roy5699 4.8K Oct  2 14:02 som3d_colors_6_2.geomodeling
-rw-rw-r--  1 roy5699 roy5699   51 Oct  2 14:02 som3d colors 6 2.CLM
-rw-rw-r--  1 roy5699 roy5699 5.3K Oct  2 14:02 som3d_colors_6_2.alut
-rw-rw-r--  1 roy5699 roy5699  20K Oct  2 14:02 fort.66
[roy5699@opal som3d_6]$
```

In visualization software like Petrel only a max of 256 colors can be imported. Thus below a second example of SOM3D is shown with a different set of attributes and with a lesser number of initial classes (256).



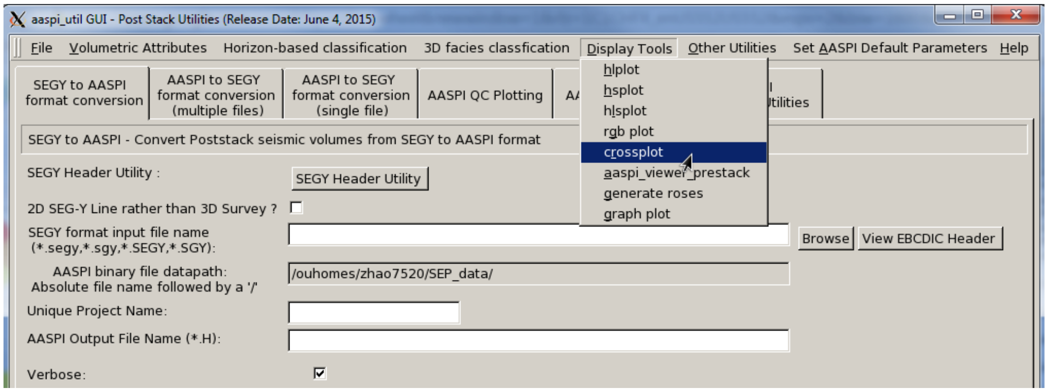
Prototype vectors projected against the first two eigenvectors for iteration 1 and 2.



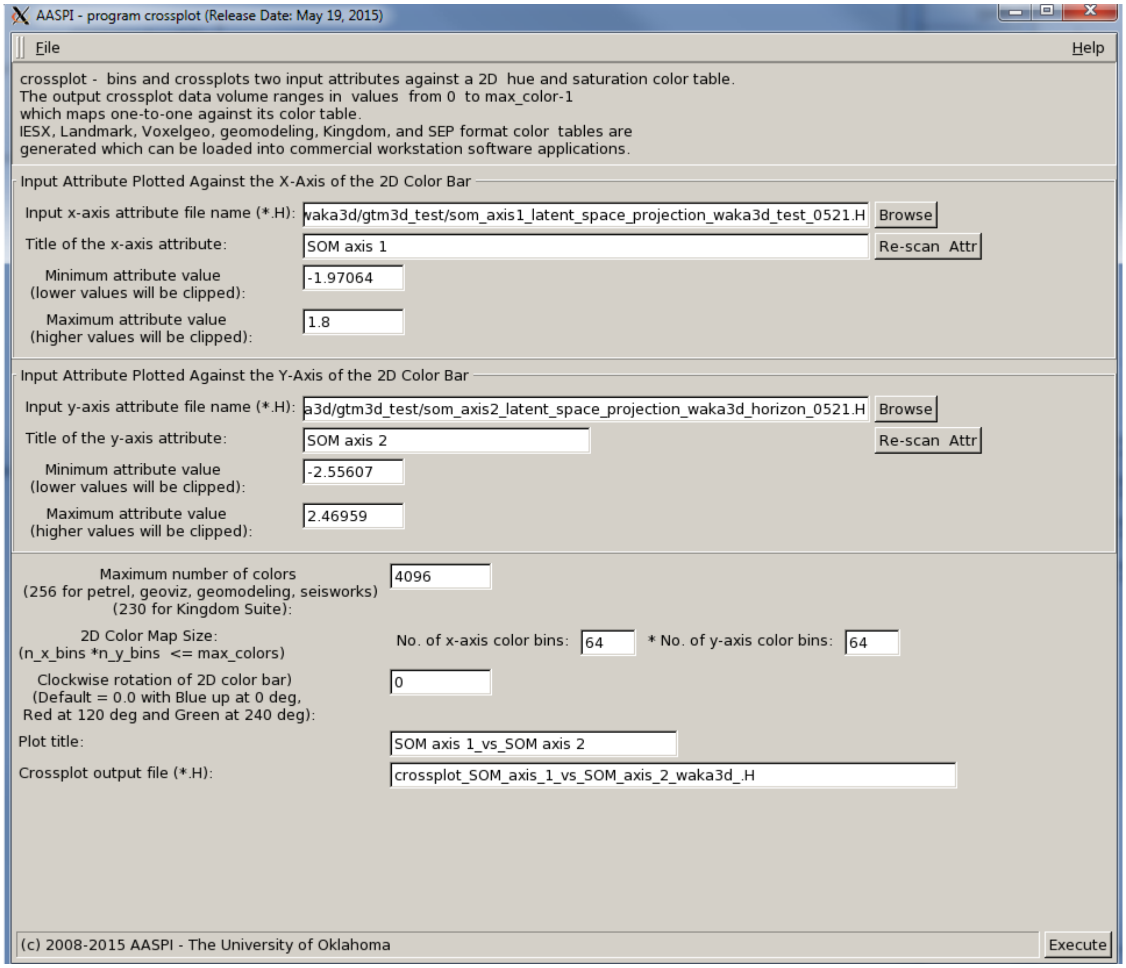
Output generated from the facies clustering module with a lessser number of classes 256. The above figure shows a representative time slice of the Oswego level at iteration 1 and 2. Different colors indicated by the arrows indicates different depositional seismc facies. Arrows indicate a channel facies which becomes more prominent after the second iteration.

Plotting (2): crossplot module

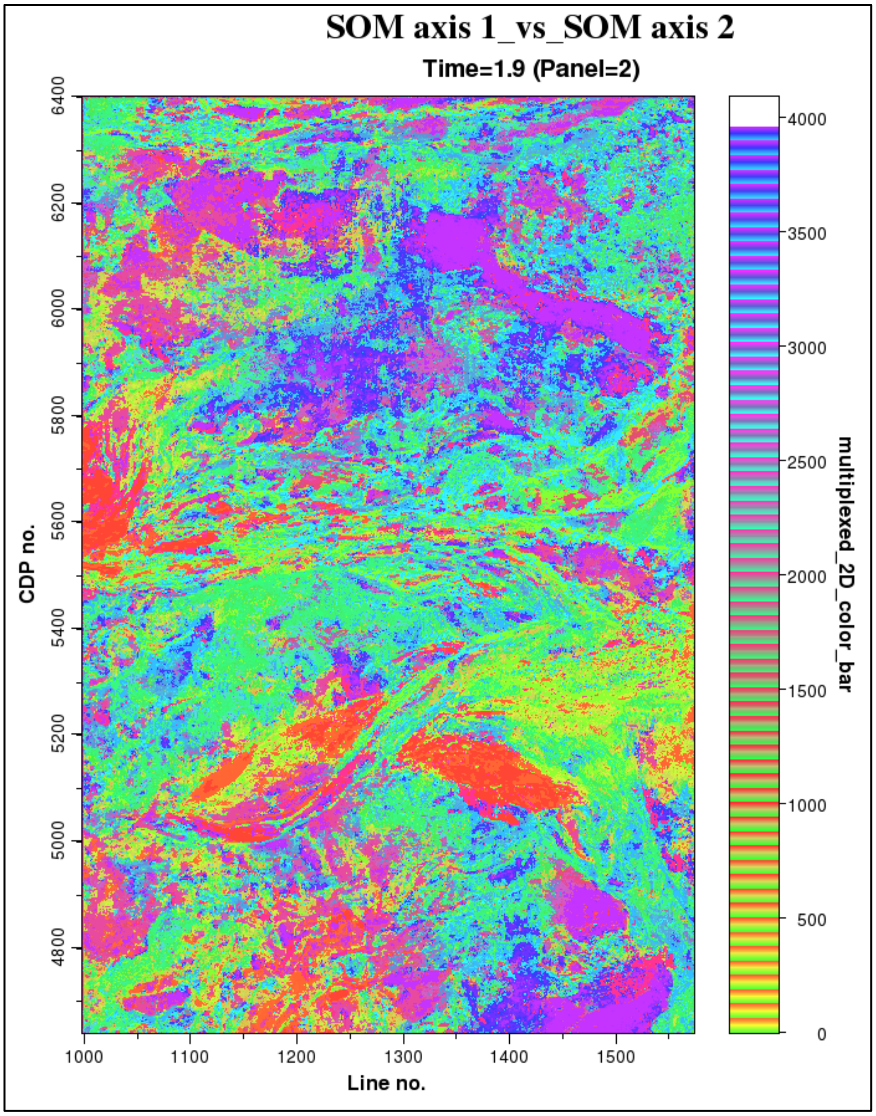
The user can also use **crossplot** module in the **aaspi_util** to crossplot two generated SOM axes, generating the same classification volume as the last iteration volume generated in the **plot_clusters** module. The **crossplot** module can be found under *Display tools* in the **aaspi_util** GUI:



The crossplot GUI is shown below:



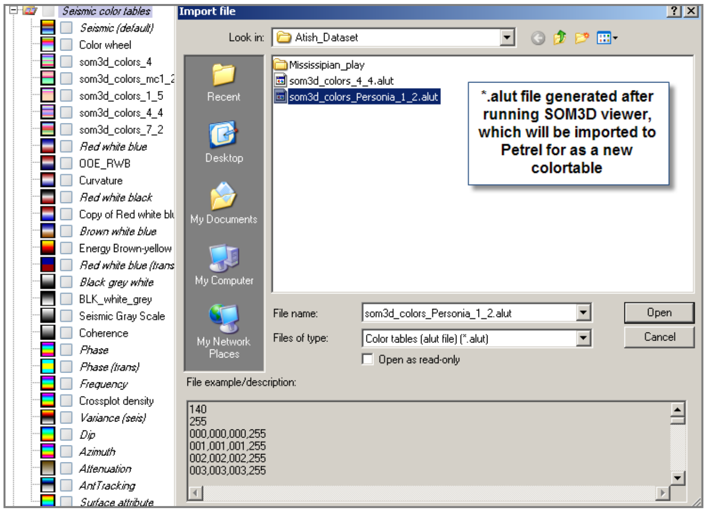
SOM axes 1 and 2 are taken as inputs for x and y axes in the crossplot. And to ensure a smooth color transition, 4096 colors are used for the 2D colorbar to be generated (64 by 64 colors). The result is shown below (from a different survey in Canterbury basin, New Zealand):



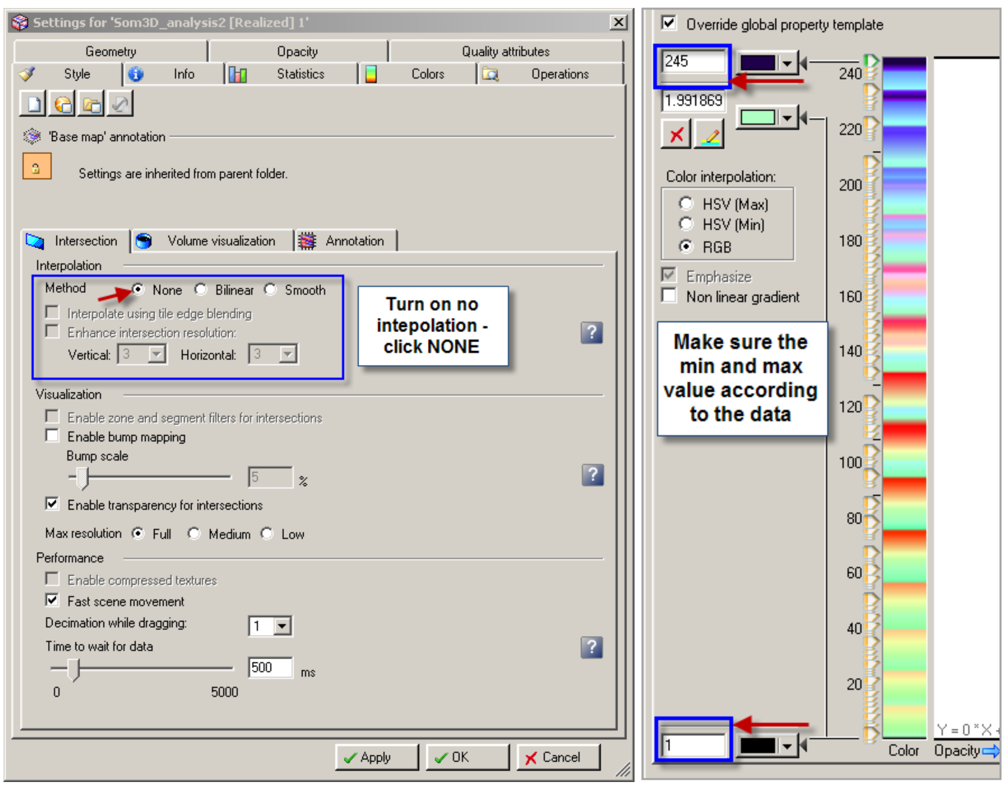
Visualization of the volumetric clustered outputs volumes in Petrel (old)

The following tutorial shows the important steps of exporting the final trained dataset and the corresponding colorfiles into Petrel and making volume probes for better interpretation. The *.segy* and the *.alut* files (highlighted in yellow arrows above section) are loaded into Petrel. This new colorbar should be used for visualizing the clustered volume.

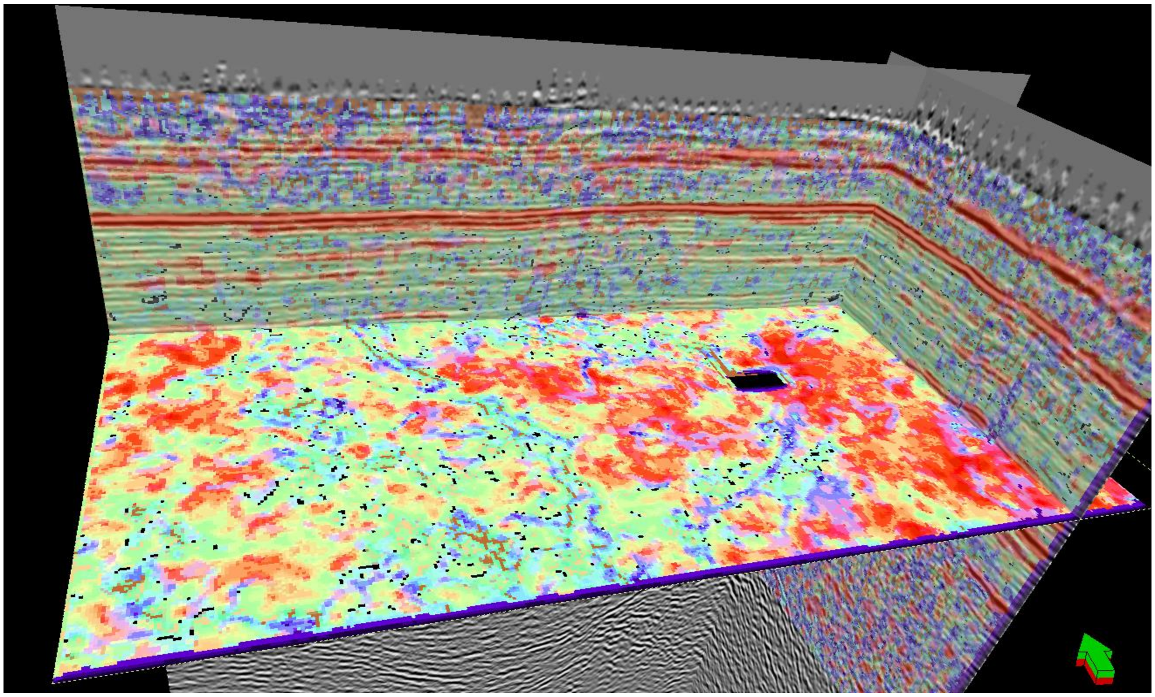
Volumetric Classification: Program **som3d**



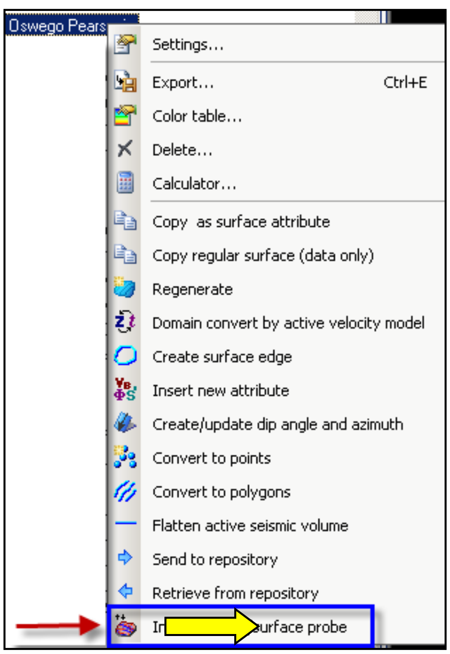
Before visualizing the volumes in Petrel the following changes in the settings should be done :



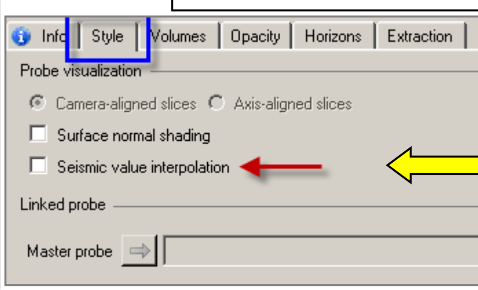
For better visualization there should be no interpolation between adjacent voxels and it should be turned off (highlighted in the blue box). Make sure the max and min range of the dataset corresponding to the ranges shown in the histogram in the settings panel.

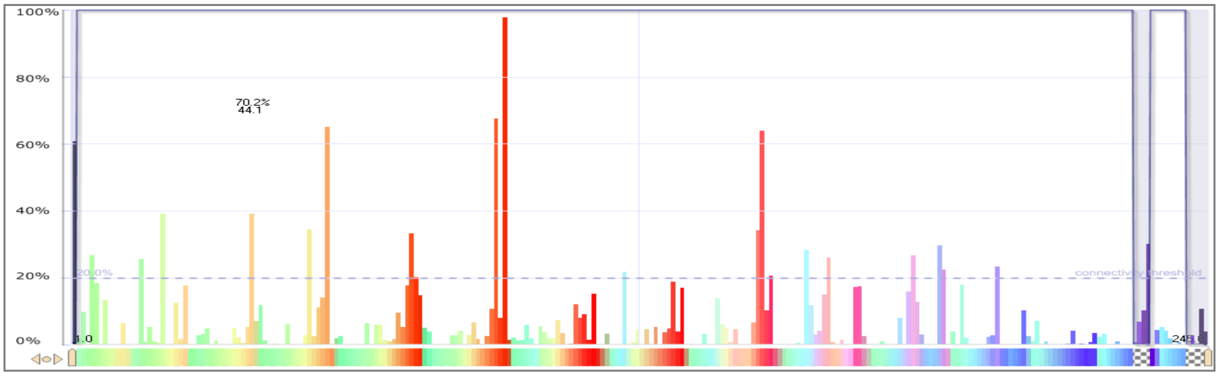


Above is an example visualization of the 3D clustering output in Petrel. The inline and crosslines are co-rendered with the seismic amplitude for better interpretation of the results.

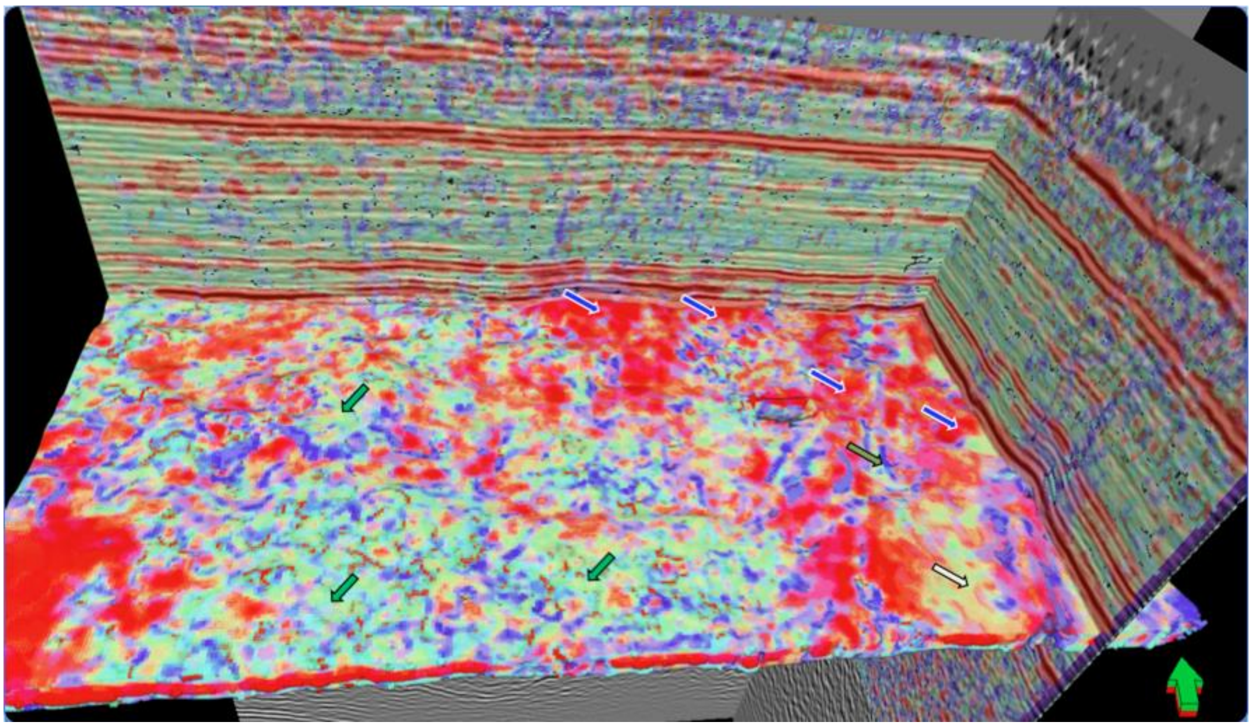


Step 3: Creating surface probe in Petrel: Right click on any horizon and – Insert horizon/surface probe for the required horizon. In the probe settings do not click on Seismic value interpolation. Click on the opacity tab to see the histogram of the input dataset.

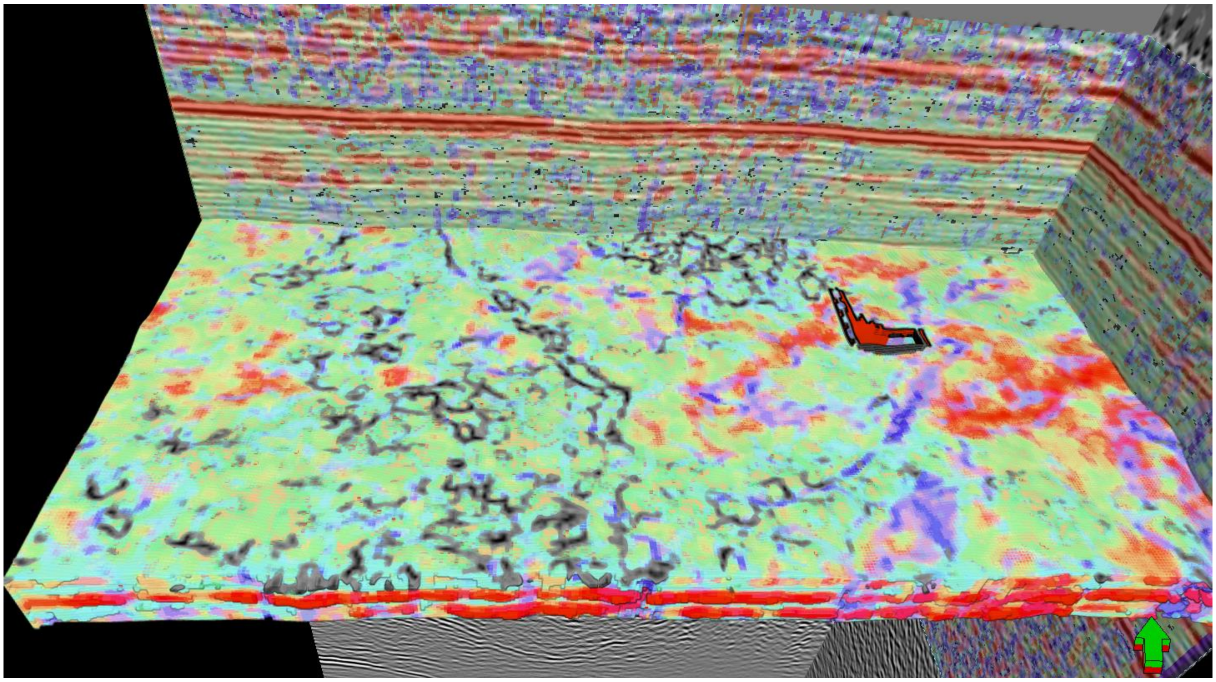




The above figure shows the histogram of the input dataset. One can play with the transparency to create different geobodies.



Above is an example of a surface probe of the 3D seismic facies volume in Petrel. The various lithological variations and discontinuities can be observed in the figure.

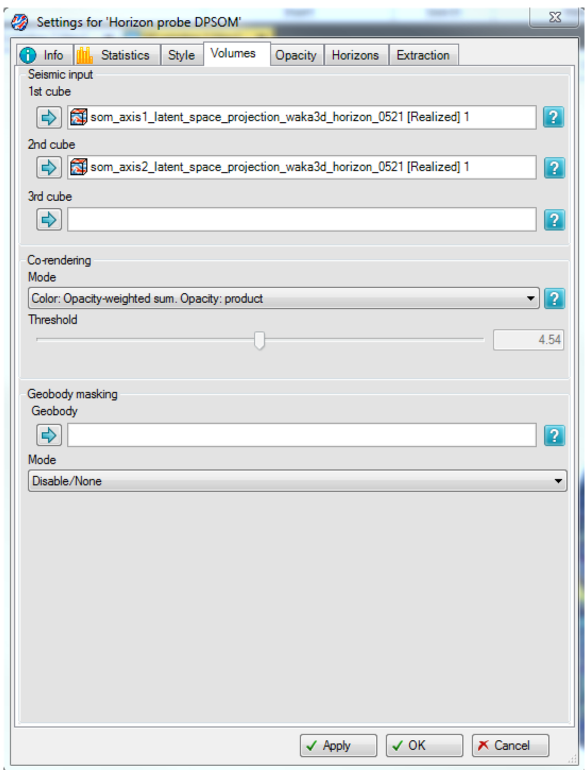


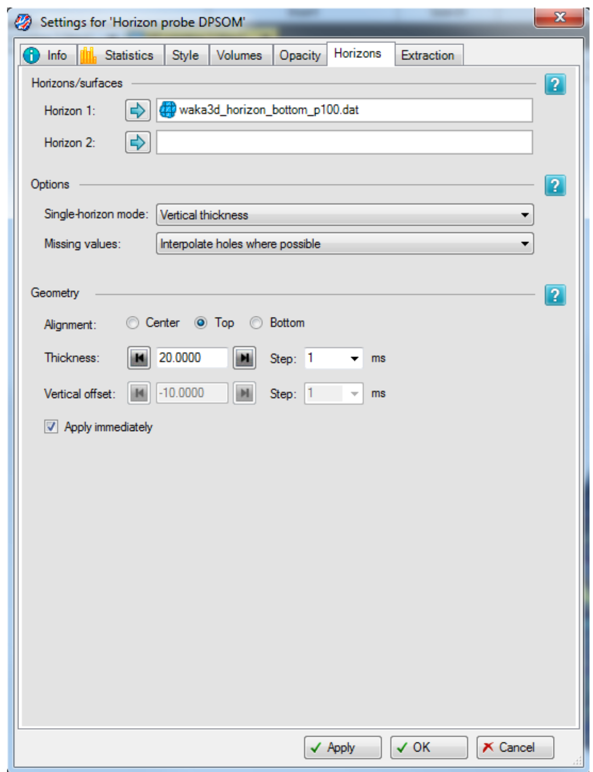
Example of a surface probe of the 3D seismic facies volume co-rendered with a coherency volume to map the channel boundaries better interpretation.

Visualization by corssplotting two SOM axes in Petrel

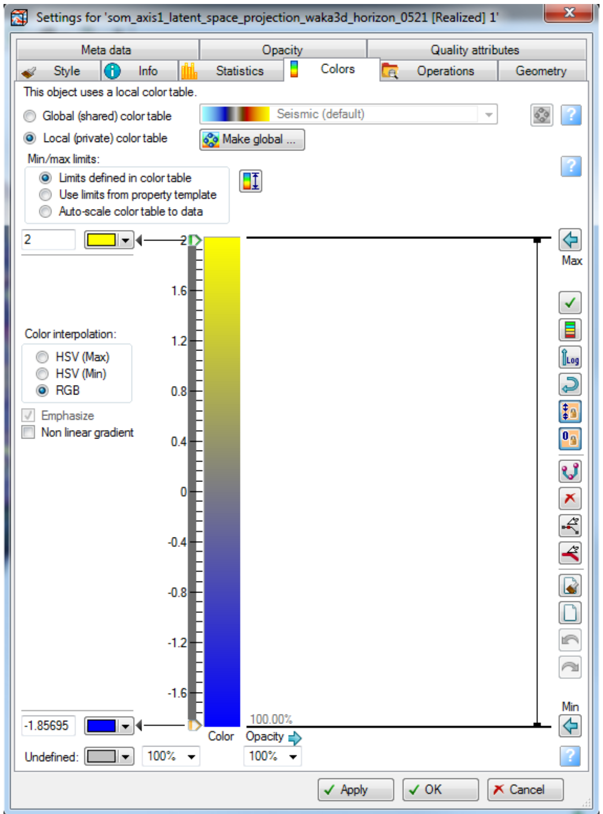
In the previous section, we described how to visualize the AASPI generated SOM facies volume in Petrel. However, as discussed in the earlier sections, most commercial interpretation software can only display a finite number of colors if using a discrete colorbar. To overcome this issue, instead of generate 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.

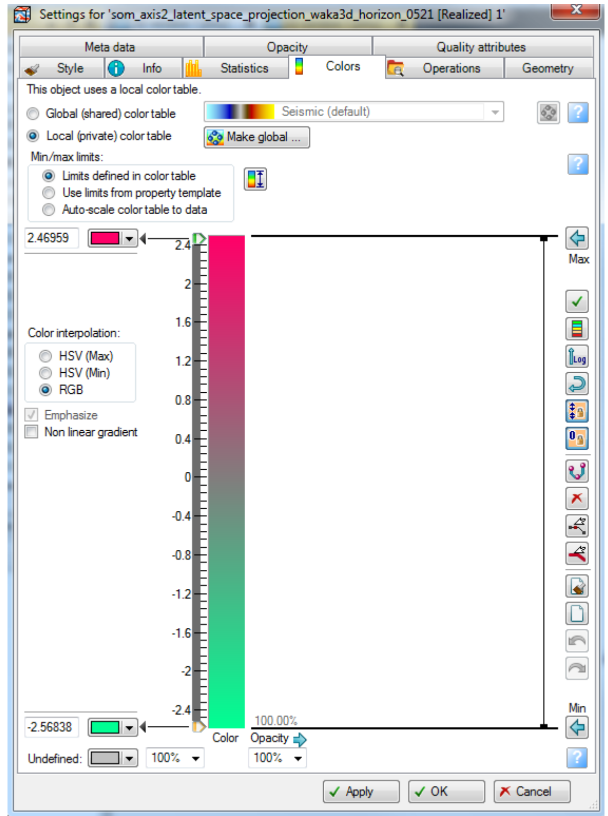
We import the two SOM axes into Petrel, and use the volumetric corendering probes to corender these two volumes. In this case, we use horizon probe as we want to display the facies along a horizon of interest. As shown in the screenshots below, we select the two SOM axes as the input volumes, and make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



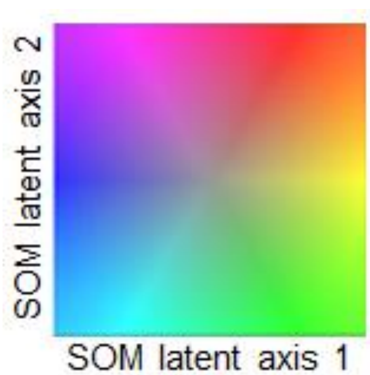


In order to fake a 2D colorbar, we then need to change the colorbars of the two SOM axis volumes as follows, and make the max and min value of the colorbar to best fit the data range:

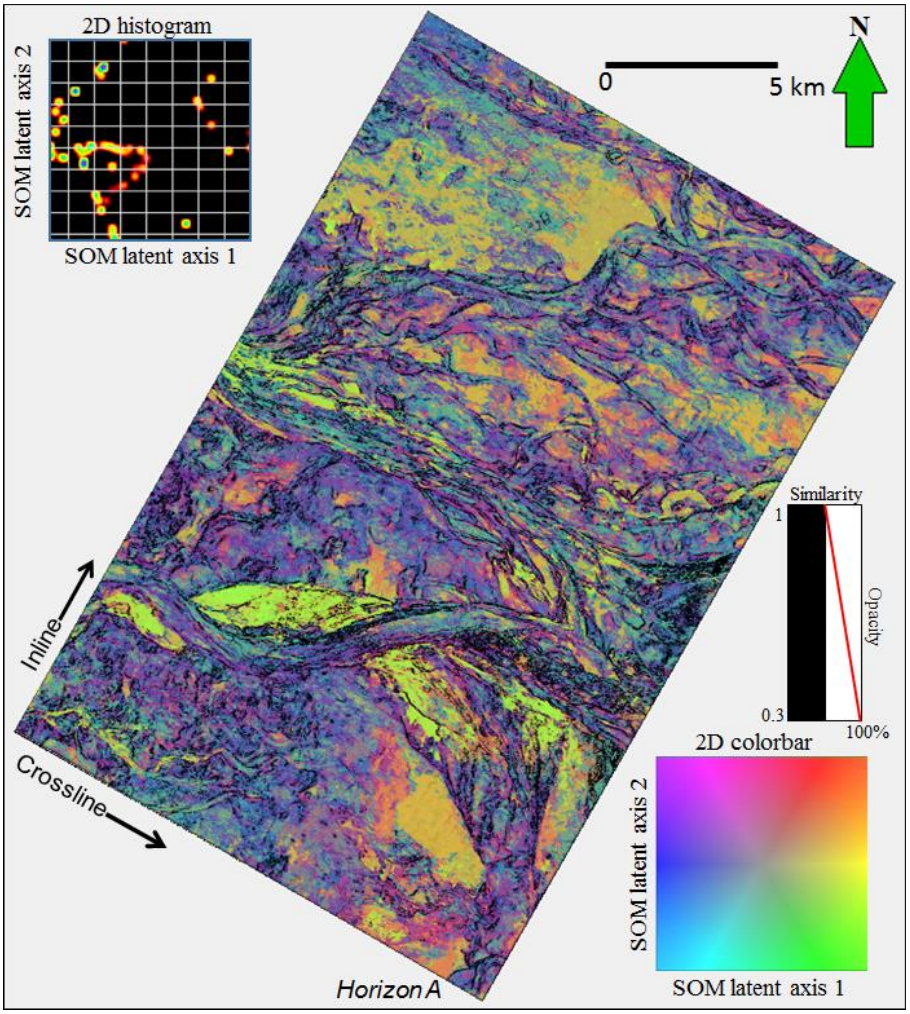




In this way, we are actually faking a 2D colorbar like this:



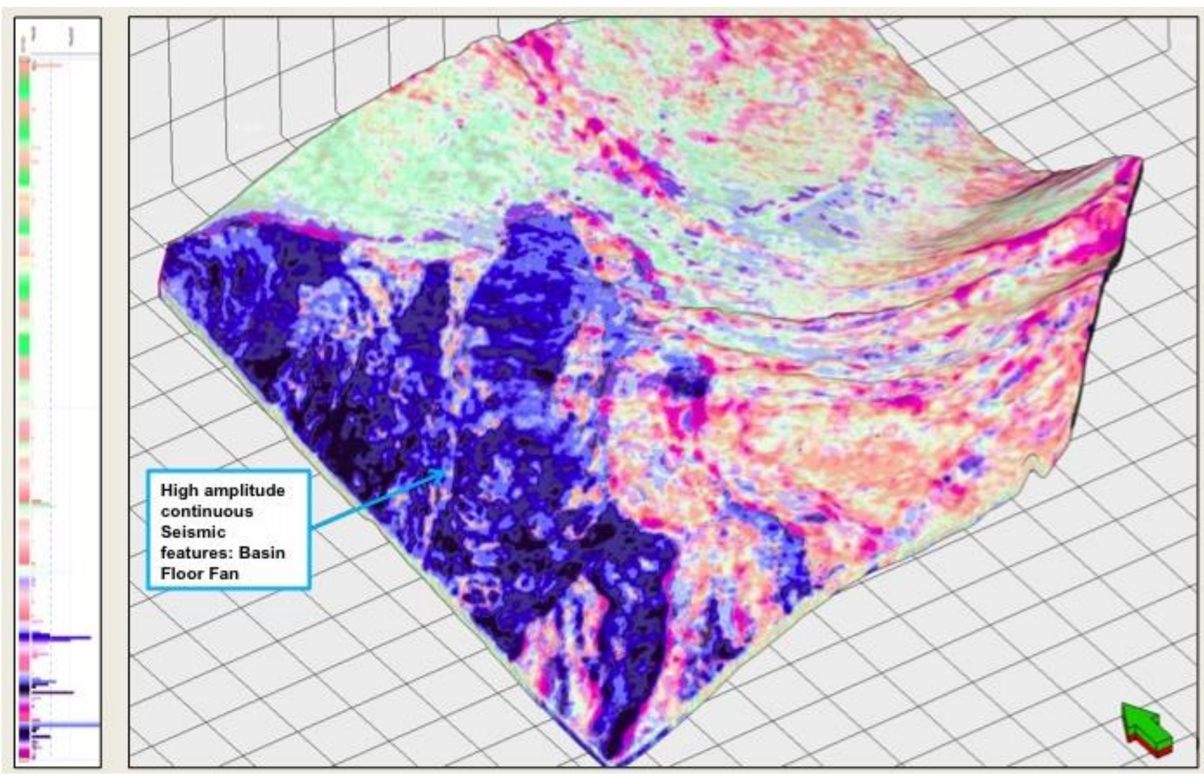
And if we see the horizon probe from top (display in a 2D window in Petrel), the crossplotting (corendering) result will be look like this:



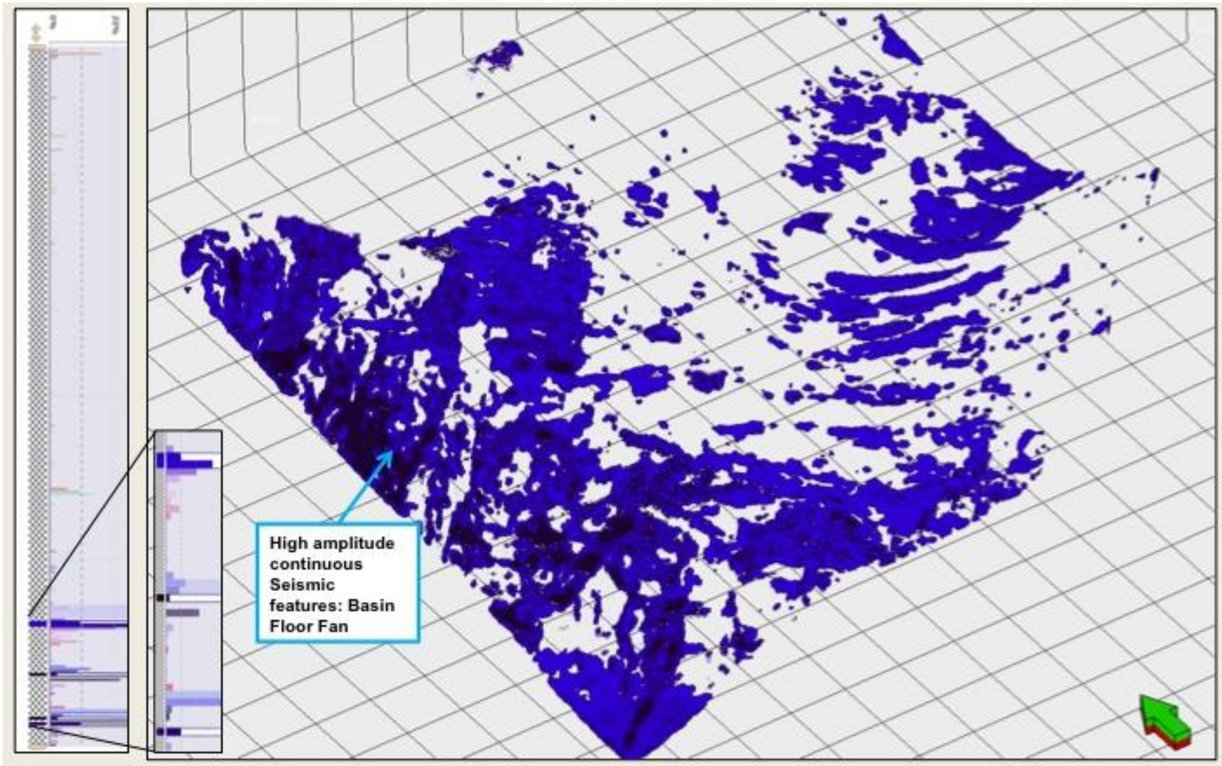
In this image we are also corendering Sobel filter similarity to highlight the edges. To do so we extracted the Sobel filter similarity along the horizon that we used in the horizon probe, and display in the same window with the horizon probe, using an opacity curve shown in the figure. The case study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

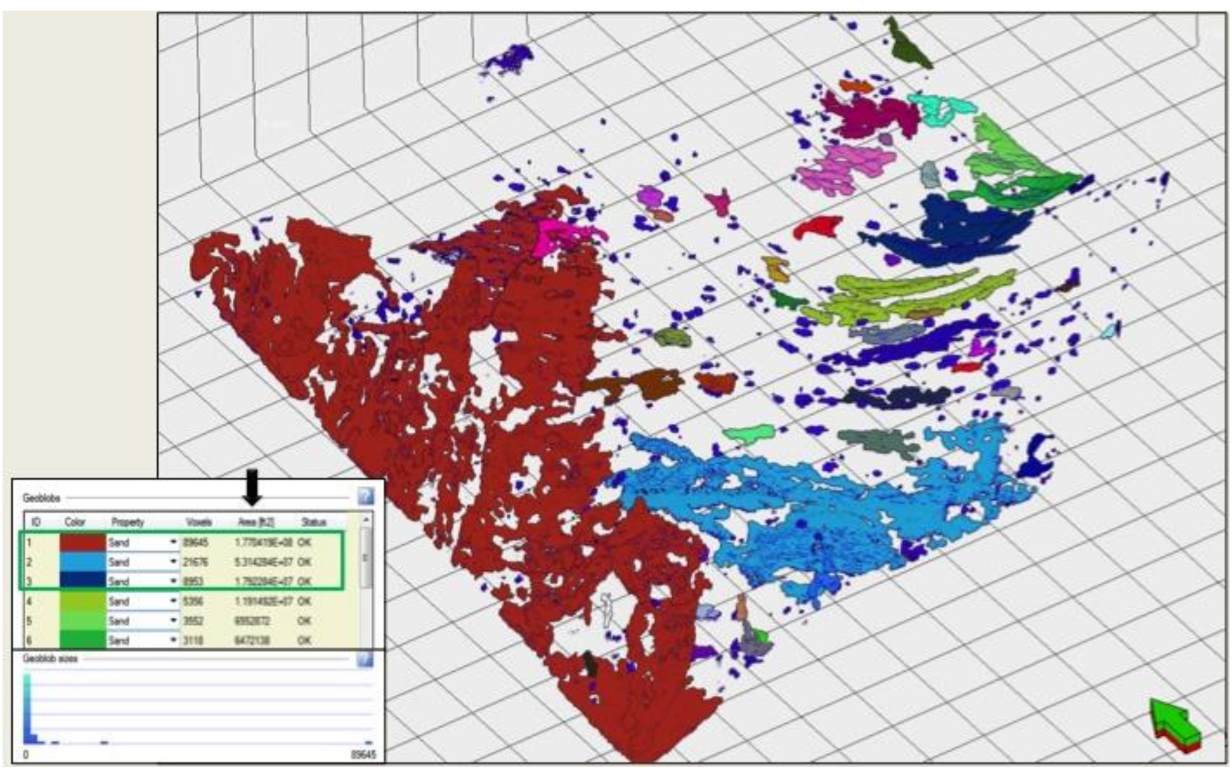
Geobodies extraction on the facies volume in Petrel (old)

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). The figure below 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). The last figure shows the output after running automatic geobody extraction in Petrel. These geobodies gives a more quantitative estimation of the seismic facies.



Volumetric Classification: Program **som3d**





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-organizingSelf-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.

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, Matos, Marcilio, 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).