Report on SBIR Phase II DE-SC000819 Research and Development: Implementation of SKATE[1], a Web-Based Seismogram Digitization Product

Retriever Technology
1600 Lena St. STE D2
Santa Fe, NM 87505
info@retrievertech.com

Table of Contents

- Introduction
- Definitions
- The Pipeline
  - Get Region of Interest (ROI)
  - Get Meanlines
  - Flatten Background
  - Ridge detection
  - Segmentation
  - Skeletonize
  - Intersections
  - Trace assignment
    - Nearest meanline assignment
  - Other Segment Assignment Methods
  - Editing
- UI and computer architecture
  - The UI
  - Terminology:
  - The Pipeline in Python
- Concluding remarks
  - Caltech archives
- Appendix A
- USER MANUAL V1.0
  - Introduction
  - Current Status and Available Images

[1] Seismogram Kit for Automatic Trace Extraction

Requirements
How to use SKATE
Operating Instructions
Definitions
Login
Searching
Browsing and Viewing
Zooming in and out
Editing
Editing Tools
Edit Mean Lines
Moving a meanline
Deleting a meanline.
Adding a meanline
Erase Segments
Segment Assignment
Editing Segment Assignment
Saving or discarding changes. Downloading data.
Discard Changes
Save Changes
Download Data
roi.json
meanlines.json
segments.json
assignment.json
assignment.csv
mSEED
Logout
Final

Introduction

Large stores of seismic information currently exist on paper, film and other media, containing unique records of earthquakes, geological exploration, and nuclear testing events. These data have significant scientific and societal value, but are currently in a form that is not available to modern digital analysis techniques. In order to become useful, the seismic data from the traces on these seismograms and other records needs to be extracted and outputted as computer readable binary data, a process referred to in this proposal as digitization.

In order to meet the need for a fast and reliable seismogram digitization tool, Retriever Technology has created a prototype version of SKATE - Seismogram Kit for Automatic Trace Extraction. While other digitization tools exist, they are not fast, require significant user

interaction, and have a difficult time handling trace crossings during periods of seismic activity. In addition, they typically require users to purchase and install software that can be expensive, and need to maintain a computer on which to process the images.

With these limitations in mind, we have created a browser-based software tool with the following attributes.

- Server based. The user need only navigate a browser to our User Interface, hosted at seismo.redfish.com. No software installation is required. No purchasing of software is required. Minimal client computer requirements because all intensive processing is done on the remote server.
- Low cost. Currently the cost to digitize a full sized WWSSN seismogram is approximately $0.02 per image. The use of remote Amazon EC2 servers allows us to utilize processing power that is out of reach for individual users.
- Reduced user interaction. We have developed a series of algorithms that separate the traces from the background and unwanted noise features. We have developed editing tools to clean up the images, and a segment connection routine that assigns traces to their proper place in the time series.

Under several contracts with the USGS, including but not limited to contract #G09PC00116, #G09PD02064, and #G10PD02236, we created scanned records of over 150,000 WWSSN seismograms. These were delivered to the USGS on removable external drives in order to fulfill contract requirements. We then used our (required) copies of these seismograms to create our own archive, and have used it to help guide our development of SKATE by using the wide variety of images to develop the best image processing techniques. This archive is available free to any user for examination, downloading of full sized images, and editing of a subset of processed images.

The SKATE website has three types of images:

- What we call ‘processed data,’ meaning that we have processed approximately 30,000 randomly selected images up to the point of automatic meanline assignment, and segment and centerline identification. These data have not been run through the final segment assignment algorithm. This assignment algorithm can be run at any time, but we feel that absent some user interaction and editing, the final result will have less than ideal results. These images were limited to WWSSN long period seismograms from 1970 and on. There is also a subset of images called ‘problematic’ which simply means that the number of identified segments exceeds a threshold value above which display in the browser becomes more time consuming. Nevertheless, this subset of processed data has good data except for a few instances.
- The second type of processed images are listed as ‘has edited data.’ This is a small subset of approximately 20 images that we have hand edited to remove spurious features and to ensure that the meanlines are properly placed. They were then run through the segment assignment algorithm and have data that is ready to be downloaded. Please refer to later sections in this manual that discuss editing and downloading of data.

- There is a third category available through the UI called ‘no data.’ There are over 150,000 of these high resolution scans available for viewing and downloading. Budget limitations due to Amazon’s EC2 operational costs prevent these from being processed at this time.

The User Manual for SKATE can be found on the SKATE home page, and describes the complete operational procedures. Users can edit, run segment connection algorithms, and download time series data on any of the ‘processed’ seismograms. Owing to the aforementioned budgetary restraints, we do not at this time allow for the processing of the ‘no data’ image types.

The SKATE website is the end product of our Phase II research, and represents the current status of our R&D work. The rest of this report will discuss the underlying algorithms and techniques that are used to digitize seismograms, followed by a discussion of the operation of the User Interface. We will not report on all of our R&D work, but only that which was incorporated into the current version of SKATE. Other research has been performed that will be part of our future developmental efforts, and much of this is discussed in our Sequential Phase IIB proposal that we are submitting in response to the DOE FOA 0001193. If successfully funded, this IIB work will allow us to incorporate existing research and develop new techniques, and will enable us to keep the SKATE website running for many years.

This report has detailed information on algorithm processes, beyond what might be required. We believe that the SKATE website itself is the best descriptor of our work so for. Nevertheless, we are using this report as a detailed document so that future work will have a source of detailed information to draw from in order to completely understand the processes used.

Definitions

The following is a list of definitions of some of the more germane terms used in this report.

Centerlines. Centerlines represent the actual path that a trace is following. It is a single pixel wide object that tracks the middle of a segment as traced by the original light beam in WWSSN seismograms..

Digitize. Digitizing a seismogram takes the scanned image file and extracts the time series that represents the seismic trace’s amplitude as a function of time. We output the time series as either a .json file or a .csv. Other formats such as mSEED will be created in future versions.

Domain. The range of x-values (time-values) that a segment traverses.

EC2. Amazon Elastic Compute Cloud (EC2) forms a central part of Amazon.com's cloud - computing platform, Amazon Web Services (AWS), by allowing users to rent virtual computers on which to run their own computer applications. EC2 encourages scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image to configure a virtual machine, which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate server-instances as needed, paying

by the hour for active servers - hence the term "elastic"[2]. Retriever Technology uses EC2 instances on an as-needed basis to digitize the seismograms. The digitization process requires a higher-cost instance. Editing the seismograms uses a lower-cost instance. All calculations and the operation of the SKATE website require EC2 use, and incurs financial cost.

Meanlines. This is the zero-energy line that a trace would follow if there was no seismic activity. They are used for subsequent trace assignment and connection algorithms
ROI. The ROI is the Region of Interest in the WWSSN seismogram image. It is the portion of the image that contains the actual seismic traces. In the image below, the ROI is shaded in light blue. The ROI can always be viewed in the UI by clicking on the Region of Interest checkbox.

S3. Amazon S3 (Simple Storage Service) is an online file storage web service offered by Amazon Web Services. Amazon S3 provides storage through web services interfaces (REST, SOAP, and BitTorrent)[3]. Retriever Technology stores all of its scanned seismograms on S3. Monthly costs accrue for this service.
Scanning. Seismograms that are scanned are simple image files, usually .tif or .png.
Segments. A seismogram image file consists of foreground and background. Foreground features are the traces themselves. Our task is to identify and separate them from the background. Background includes all non trace features such as noise, and additional features such as hand written notes, etc. Because of timing marks, trace crossings, and the multiple hourly lines found in a WWSSN seismogram, the traces are broken up into pieces. We call these pieces of a seismic trace a segment.
Trace. A trace is the actual two dimensional path created by the light beam in WWSSN images, and is the feature in the image file that we are extracting and using to calculate actual time series data.

[2] https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud
[3] https://en.wikipedia.org/wiki/Amazon_S3

The Pipeline

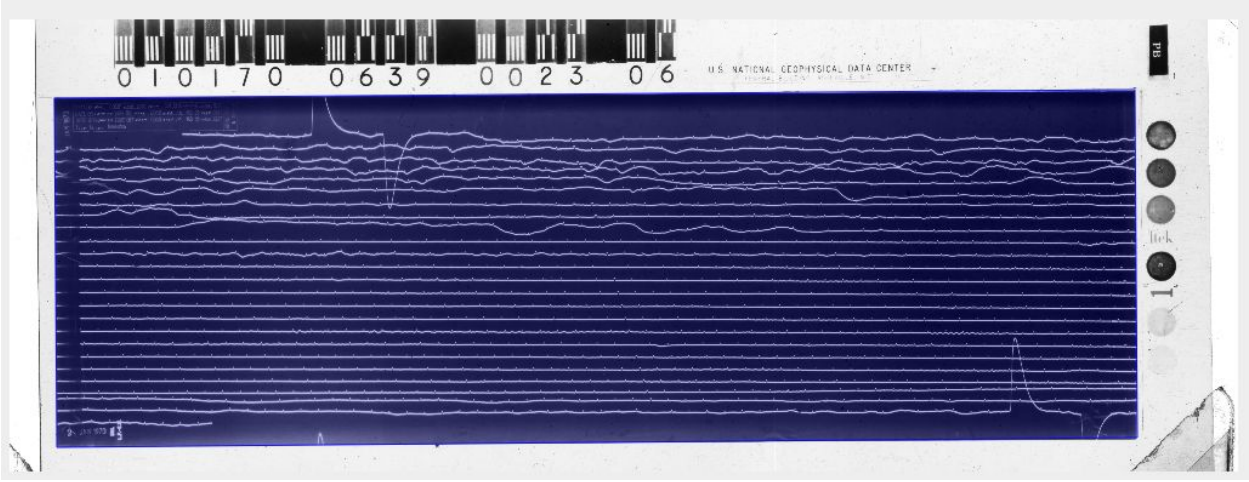
The core of our work is the extraction of trace time series data via an image processing 'pipeline' that is easily integrated, at no cost to the end user, by utilizing the free and easily deployed Python[4] programming language. Python has extensive libraries of image processing functions. We have created a comprehensive set of modules written in Python, utilizing a large library of existing image processing functions, that process the image and extract and output centerline, i.e. timeseries, data.

The pipeline is currently optimized to process WWSSN long period images. The decision to focus on this particular set of data allows us to develop generic processes that can be optimized for other types of images, including short period WWSSN images, as well as others[5]. Future versions of the pipeline will allow selection of the seismogram type, such as WWSSN short period seismograms, as well as other types of seismograms and partial images. The pipeline runs the following modules in order. Details of each module are subsequently discussed.

Get ROI → Get Meanlines → Flatten Background → Ridge Detection → Segmentation→ Skeletonization→ Intersection Detection→ Trace assignment.

Get Region of Interest (ROI)

All WWSSN images used in this work were scanned images of microfiche chips, which in turn were photographs of the original seismograms overlaid on a background that contained the file ID. A representative image is shown in Figure 1.



[4] https://www.python.org/

[5] The UI, located at seismo.redfish.com, also has examples of processed short period WWSSN images, to demonstrate this. An example later in this report also shows applicability to other image types, specifically those found in the Caltech archives at http://ds.iris.edu/seismo-archives/projects/caltech_archive

Fig. 1 ROI of a seismogram image

The ROI is shaded blue in this image. An algorithm was developed to separate the ROI from the rest of the image; without this separation, all subsequent operations would fail when attempting to analyze non-seismic information that is otherwise in the predominantly white background[6].

The algorithm proceeds as follows.

- Threshold the image using Otsu's method, and create a binary image.
- Perform a morphological opening to remove the traces which would otherwise confound the analysis.
- A connected components analysis using find_boundaries in Python was used to identify all feature boundaries. The largest feature is the ROI of interest. The expected ROI area is very consistent across all seismogram types, and was used as an accuracy test to ensure that a reasonable area was detected.
- A Hough transform[7] analysis was used with appropriate parameters to identify the horizontal and vertical edges of the ROI, using the perimeter of the largest connected component object. The intersections of these lines defined the corners of the ROI, which were stored as .json features.

The algorithm is very accurate. It tends to be somewhat conservative in its results, such that traces at the edge of the ROI are rarely cut off. Later editing tools allow for the ROI to be edited as necessary. We also will develop tools that allow for nonlinear ROI boundaries, to more accurately reflect the boundary.

Get Meanlines

Meanlines are the zero-energy lines about which the seismic traces oscillate. For a WWSSN long period image, there are typically 24 meanlines in the image; one per hour, traversing the width of the seismogram. One simple way of looking at their role in the segment assignment solution is to note that if all trace segments are assigned to a meanline, then the analysis is complete. This idea is complementary to other techniques that we use; for example, in future work we also will incorporate trace priors to attach previous to subsequent segments. We utilize both of these concepts - assigning segments to meanlines and assigning segments to other segments - in our current and ongoing trace connection algorithms.

The meanline detection algorithm uses a Hough transform to determine meanlines from the traces themselves. The image is thresholded using Otsu's method, creating a binary image, and an morphological opening operation is used to eliminate timing marks. Other work found that timing marks were not as accurate in determining meanlines.

A Hough transform is performed on the binary image, with parameters limiting the search to a narrow angular range. The resultant lines are plotted over the full width of the image and saved as a .json file for later use.

---

[6] We have also developed automatic ID identification, using hit or miss techniques to identify the actual ID numbers on the image. This was not used in this work because file names are already associated with the image.

[7] https://en.wikipedia.org/wiki/Hough_transform

Accuracy is very good, though there is still room for improvement, especially with active traces and low contrast images. A histogram of the number of meanlines found over a set of 100 random images is shown in Figure 2.

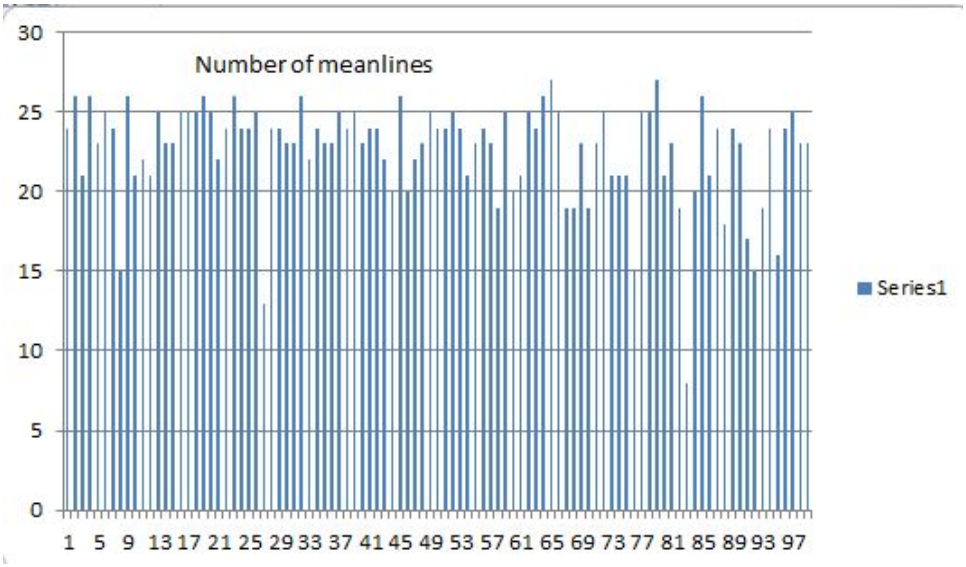


Fig. 2 Histogram of meanline counts across 100 random images

Ideally all seismograms should have 24 lines, but mid-hour start and finish times, and seismograms that do not encompass the entire 24 time span will affect the actual number of meanlines found. The editor, discussed later, allows meanlines to be added, deleted, and moved. It is not important for our work to have exact meanlines positions at this point in our work[8], as they are used only as a baseline from which trace connection algorithms can proceed. Future work will use centerline information in data space (as opposed to much more cumbersome image space analysis) to accurately determine point by point meanline locations. This information will be used to very accurately map meanline values over the entire image.

An image with meanlines overlaid (color coded to show associations with associated segments, used in the editing process) is shown in Figure 3. Note that unevenly spaced meanlines are common.

[8] Image distortions, for instance, can lead to bowed meanlines.

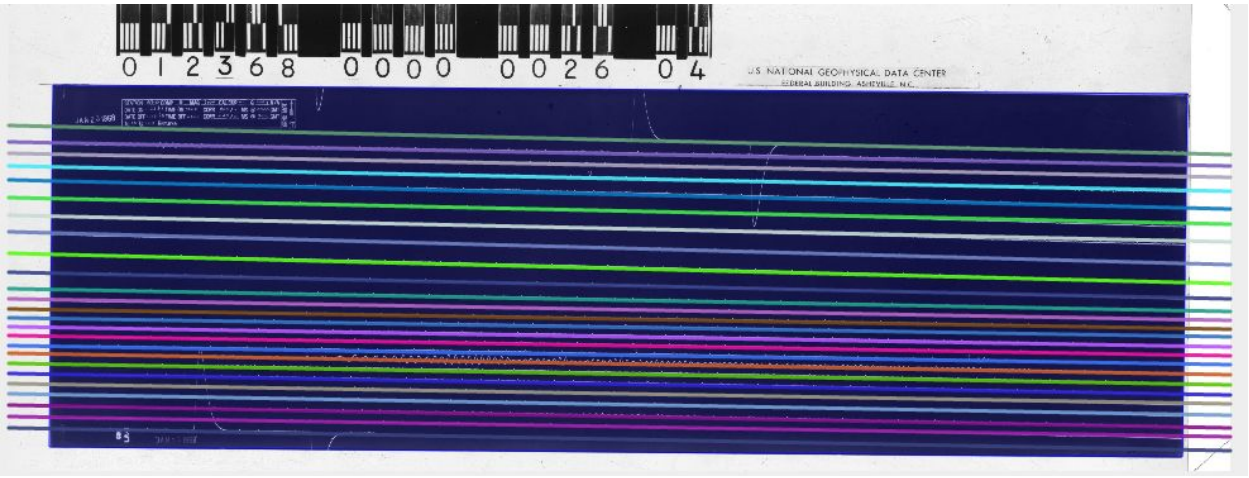


Fig. 3 Computed meanlines

Flatten Background

Background flattening serves several purposes. Primarily it removes low intensity background noise, both low and high frequency, which could otherwise lead to spurious feature detection[9]. It also is an important preliminary step in the segmentation of the image, that is, the separation of foreground (trace) features from background (non-trace) features. The identification of features as definite background also creates 'seeds' for later watershed segmentation algorithms that are used in the segmentation process.

The flattening process sets a smoothly varying threshold from an image by applying a threshold function to multiple randomly positioned positions within blocks of the image and using a 2-D smoothing spline to set the threshold across the image. A threshold that varies across the image is effective in dealing with images that have low frequency intensity variations due to, e.g., film processing, storage, or other non-ideal conditions. The use of a spline fit for threshold values ensures that any threshold changes are small and won't lead to unwanted artifacts.

The image is divided into blocks, with typical block size around 500 x 500. This is a reasonable value that is neither too fine nor too coarse. Block centers are selected using Mitchell's best candidate algorithm[10], which produces more even coverage of an array than a random sampling.

The order of the spline fit that is used to smooth the background is determined. The fit order is set to 3 unless a small number of blocks are used. The Scipy function SmoothBivariateSpline is used to create a spline fit for smoothing the threshold value across the image. A threshold value for the block identifies a value for pixel intensity below which pixels are part of the background with at least a 'prob_background' estimated probability. This works

[9] Since key parts of trace identification rely on local intensity changes, such as Canny edge detection algorithms, small changes in intensity at low intensity levels does lead to the identification of spurious features as real signal.

[10] https://www.khanacademy.org/computer-programming/mitchells-best-candidate-algorithm/5901958850543616 One disadvantage of this technique is that it is non-deterministic, and can make debugging somewhat more difficult.

by assuming that most of the image is dark background, that the single most common pixel brightness is the average brightness for a background pixel, and that the brightnesses of background pixels are described by a normal distribution.

Once the threshold is determined for each block, the identification of background pixels is performed. A histogram is created of all pixels for the grayscale image in each block. The expected distribution of the background is calculated by finding the mode of the dark pixels, assuming a normal distribution of background counts about the mode, and 'flipping' the distribution to the left of the mode and superimposing it on the right of the mode. This assumes that all bins with intensities less than the mode are all background. The difference between the bin values of the flipped histogram and the original histogram represents those # of pixels that could be from the foreground. A probability value is set = 0.95. This means that at some bin value (and only bin values less than 127 are considered, since we are only looking at dark/background pixels and will assume that they are all less than 127), if 95% of the pixels at that bin value are less than the 'flipped' (i.e. normal) histogram value, then _all_ pixels of that bin value are listed as background pixels. The background is flattened by setting all identified background pixels to the previously determined threshold value in that block. This eliminates unusually dark regions, and removes a majority of spurious background noise. A typical histogram of an image is shown in Figure 4. Note the tail on the dark region's higher bin values compared to the near normal distribution for those values less than the mode.

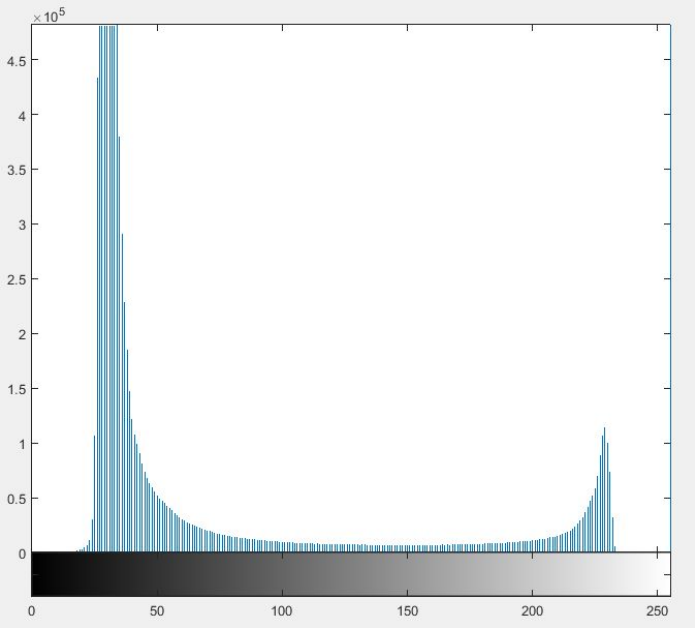


Fig. 4 Typical histogram showing intensity distribution of a seismogram

An example intensity profile of a dark region before flattening is shown in Figure 5.

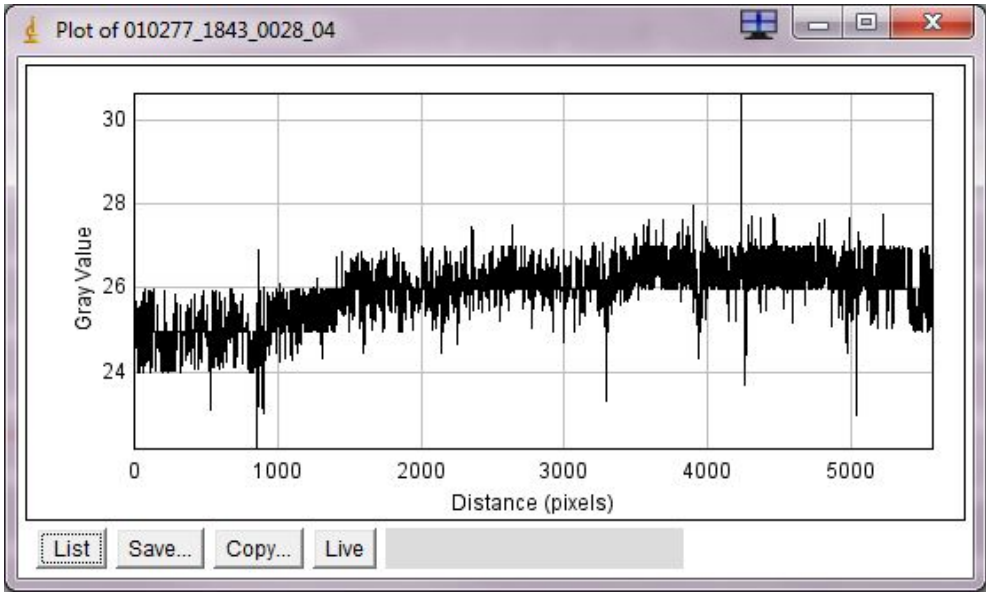


Fig. 5

And the same region after flattening is shown in Figure 6.

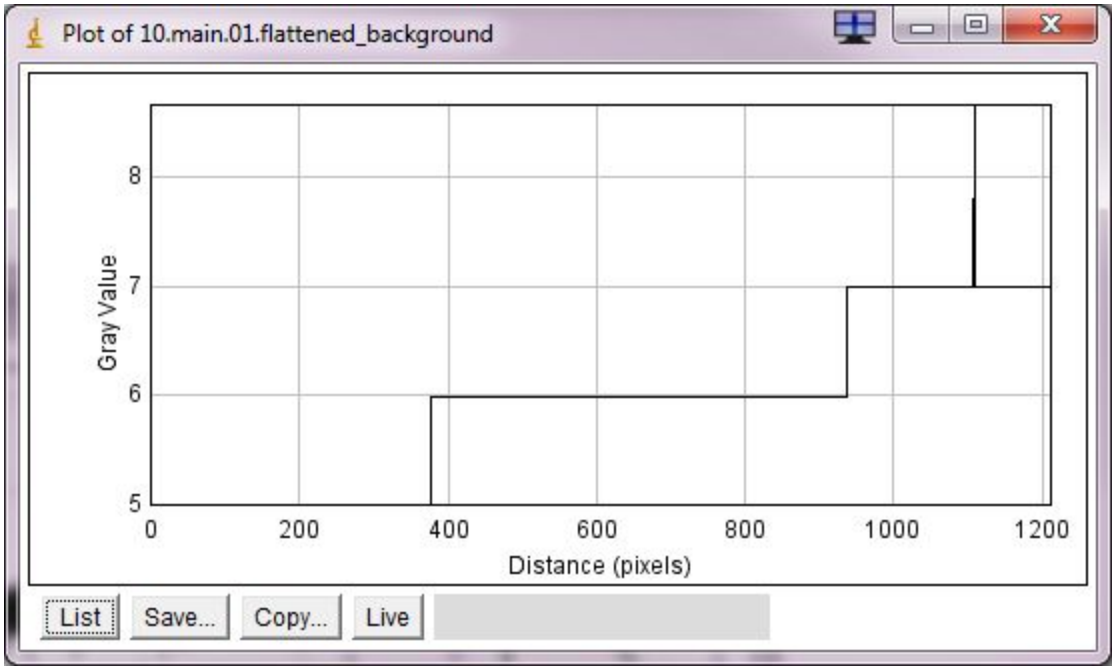


Fig. 6

And the slow changes in background intensities can be seen in the windowed image in Figure 7 of an entire seismogram.

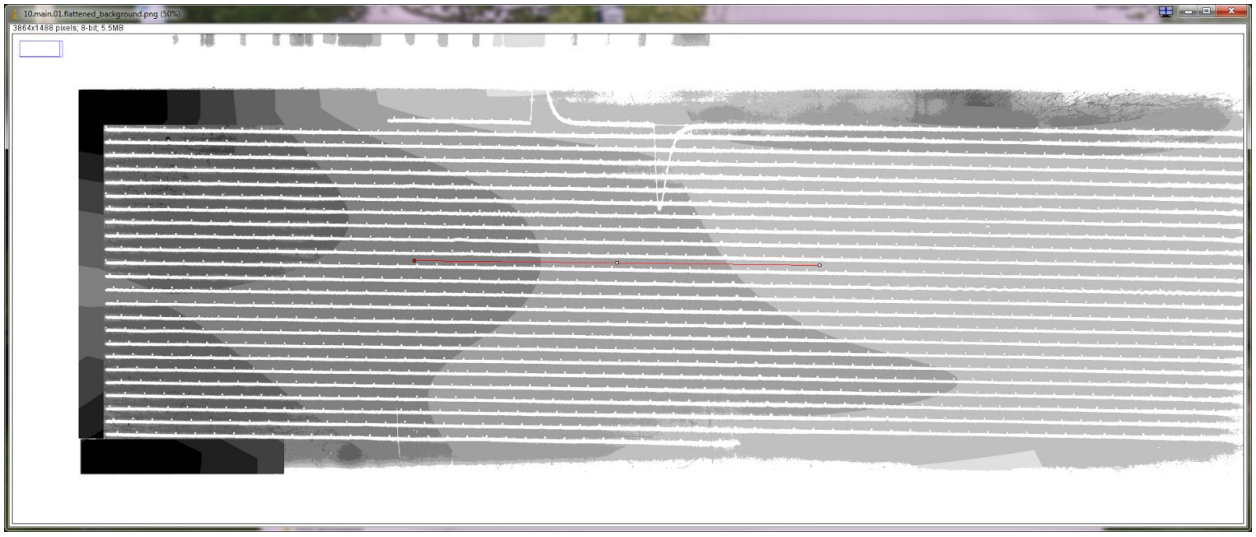


Fig. 7

Ridge detection

This module is used to identify the centerline of any given trace, using a combination of intensity based image processing tools, particularly those involving the first and second derivatives of the trace intensity across its width. Because we are seek to find the centerline of the trace, we apply a variety of convolution kernels to identify those portions of the image that both are and are not likely part of the centerline. Then, applying a variety of Boolean AND and OR operators, we combine the identified features to locate those which are most likely the centerline of the trace. In our nomenclature we call the centerlines 'ridges.' We note that it is not correct to use the maximum value of intensity across a vertically sampled column of trace intensities, as other methods use, since we have proven that seismic activity and varying trace widths will result in incorrect centerline positions.

Ridge detection is now described. First, the image is processed by a Laplacian filter. Laplacians are edge finding convolutions, and calculate the second derivative of the intensity profile of the image. For a seismogram, a typical Laplacian is shown in Figure 8. It is an 8-bit gray scale image.

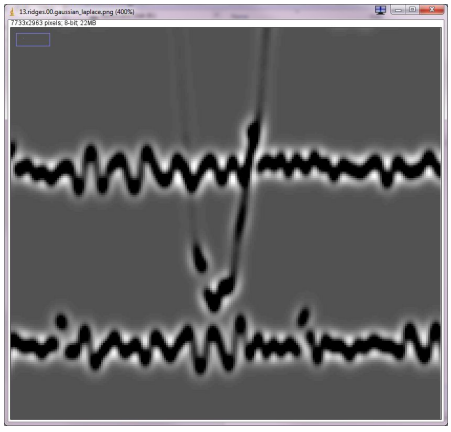


Fig. 8 Typical Laplacian image of a seismogram

The Laplacian image is then thresholded with only those pixels having a Laplacian value greater than the 'convex threshold' being kept. This threshold is a small but positive number (typically convex_threshold = 0.00015) and limits the image to only those pixels with a positive second derivative, which eliminates most background pixels. The image returned is a binary image shown in Figure 9, with only those pixels that are 'true' returned as having a +1 (i.e. white in the image) value.

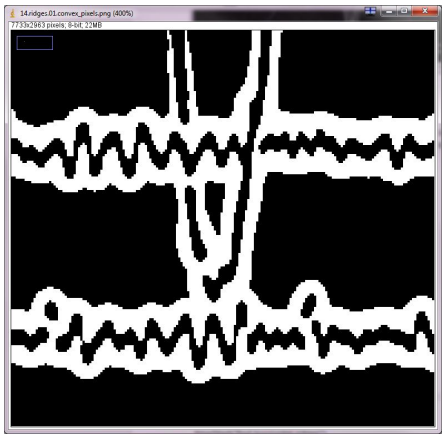


Fig. 9 Binary image with only positive 2nd derivative values kept.

This operation defines which pixels belong to the edge of a trace. All other non-edge pixels are discarded. Note that this discards both background pixels as well as trace centerline pixels. It is this feature that we will later exploit to define the trace centerline.

A challenge is to find the centerline of all traces over a wide range of intensity values. Quiescent traces have a high intensity owing to the relatively long residence time of the incident light beam on the original photosensitive recording paper[11]. Active traces are necessarily dimmer, and consequently more difficult to segment by simple intensity thresholding routines. Therefore, a core part of this module uses an edge finding algorithm, the Difference of Gaussians (DoG). The DoG is an approximation of the Laplacian of the Gaussian (LoG) filter, but is a faster implementation that gives similar results. The DoG filter allows us to find trace features over a wide intensity range while eliminating background noise. By creating an image pyramid at successive levels of blurring we can find trace features over a wide range of scale and intensity. A 1D gaussian is used separately in the horizontal and vertical directions to detect oriented features. The maximum and minimum sigma values in the pyramid were chosen to scale with the approximate widths of the traces that we are analyzing.

Preliminary calculations determined the scales at which to compute the Difference of Gaussians. The number of scales at which to compute a difference of gaussians (DoG) is set

[11]Eastman Kodak Company, "Kodak Linagraph 480 Paper,' Kodak pamphlet ; no. P-76, Rochester, NY: Kodak, 1970. Interested readers can order a copy from the Florida State University Strozier Library - General Collection.

by the low and high sigma values, and the sigma ratio[12]. For most of our work this results in an 8 level image pyramid. When processing the 30,000 images that are available on SKATE at seismo.redfish.com, however, we used a 6 level pyramid in order to speed up the process and reduce compute costs by 50%. This gave satisfactory results at this time, but it can be further improved. Nevertheless, it found features across a broad range of intensity levels, while eliminating high frequency noise contributions that could otherwise be included as real trace features.

Figure 10 below shows a DoG from a lower order value of sigma, in the vertical direction.

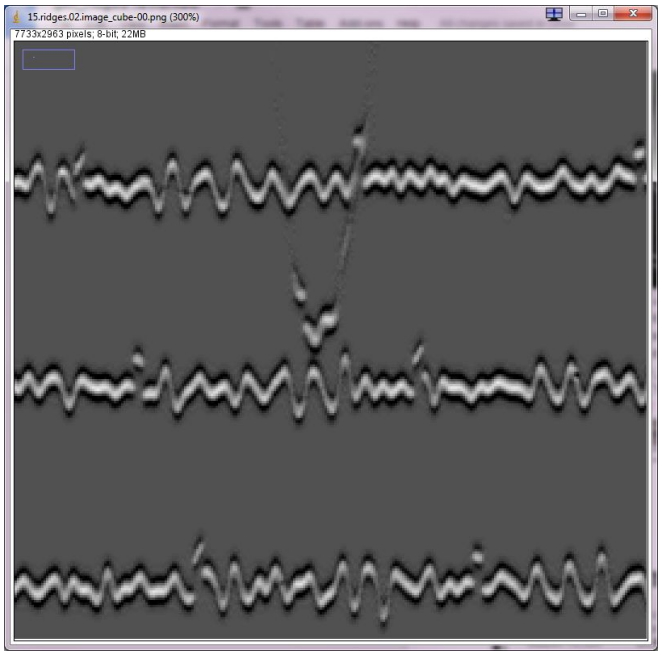


Fig. 10 Sigma level 2-1 DoG image.

And Figure 11 is a higher DoG differential image, also in the vertical direction.

[12] DoG and also SIFT references can be found at, e.g. https://en.wikipedia.org/wiki/Scale-invariant_feature_transform, http://www.cfar.umd.edu/~fer/cmsc828/classes/tutSIFT04.pdf, and a key paper by Lowe, http://www.cse.unr.edu/~bebis/CS491Y/Papers/Lowe04.pdf

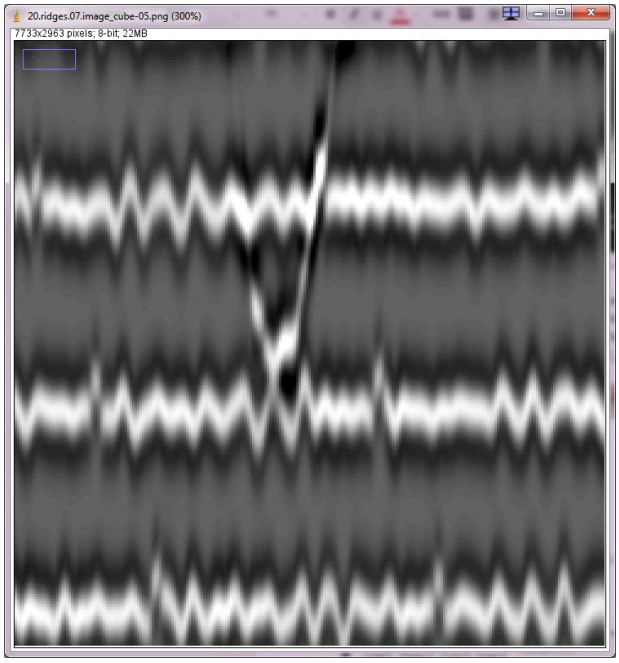


Fig. 11 Sigma Level 6-5 DoG image.

Compare Figure 11 above, which uses a vertical filter, with the image in Figure 12 of the same region and Sigma level but filtered in the horizontal direction. The directionality of the result is used later in the process to limit the search for features that are found adjacent to one another, as a means of identifying real trace features in the presence of abrupt shifts in direction, especially at intersections. All images, at various blur levels and in the two directions, are saved for later calculation.

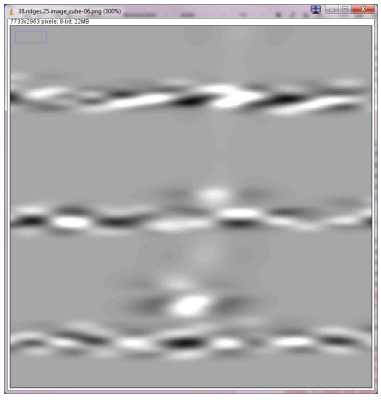


Fig. 12 The same region as Fig. 11 above, but with the 1D DoG filter in the horizontal direction.

Next, a 1-D Sobel filter in the vertical and horizontal directions, respectively, is applied to the original image. We calculate the absolute value of the Sobel to preserve all trace edges as positive values. The image is thresholded at the Otsu threshold of the sobel image, which filters the image to preserve only the steepest slopes on the traces' intensity profile. This is a way to limit the number of pixels associated with a trace. The image is saved as a binary with the steepest slopes only having a 'true' value. Similar to the Laplacian, this is a way to determine

the edge pixels of a trace, which will later be eliminated as part of the trace centerline finding algorithm.

A typical result of this process, is shown in Figure 13 for a vertical Sobel filter. Note the preference for detecting active, i.e. more vertical, features.

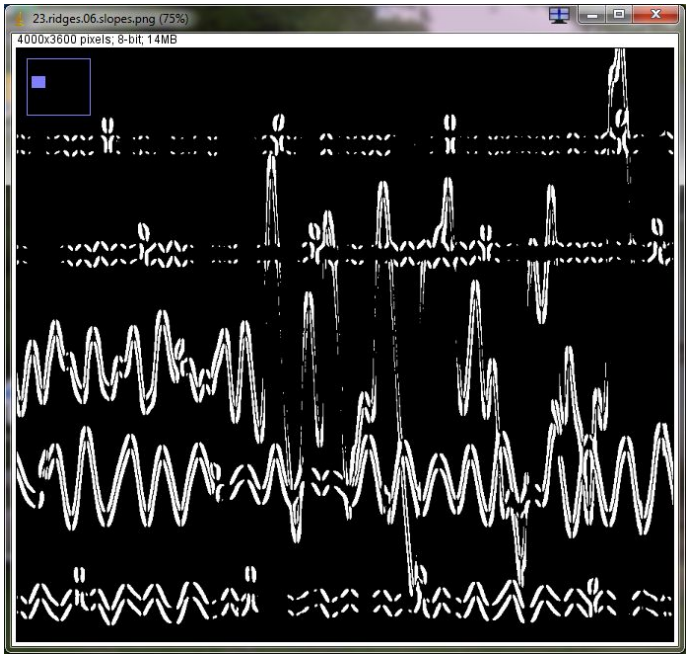


Fig. 13 Vertical Sobel kernel applied to an image

With the above series of images - Laplacian, Sobel and DoG - a bitwise OR is now performed on the following series of images: the previously determined dark_pixels from the flattening operation, OR the above defined convex_pixels, OR the steep slopes, OR the first level of the DoG pyramid, whose values are greater than the convex threshold.

The returned image, called exclusion_cube (level = i), is another binary with those image features most like to be associated with the trace assigned as TRUE, and the excluded background regions assigned as FALSE. Keep in mind that this is a segmentation process whose goal is to definitely determine those pixels which are part of a trace, while operating in an environment where noise and wide intensity ranges of true features are present. The exclusion cube for the level=1 is shown in Figure 14.

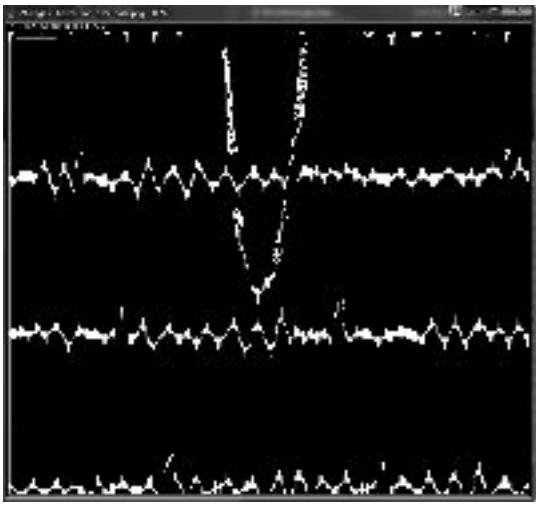


Fig. 14 First level exclusion cube

A series of images is created by performing a bitwise OR on the ith layer of the DoG pyramid with the ith-1 layer of the exclusion cube. In our case, this typically results in an 8 level series of images. Figure 15 below is i=8-7 level of the exclusion_cube.

Fig. 15 The i=8 level of the exclusion_cube

Local maxima of the previously determined image cube are found using the peak_local_max function found in scikit-image[13].(remembering that the image cube is a 3D stack of images whose fundamental construction is based on the DoG image pyramid at increasing levels of blur). The search for peak local maxima returns images that are TRUE

[13] HTTP://SCIKIT-IMAGE.ORG/DOCS/DEV/AUTO_EXAMPLES/PLOT_PEAK_LOCAL_MAX.HTML. The peak_local_max function returns the coordinates of local peaks (maxima) in an image. A maximum filter is used for finding local maxima. This operation dilates the original image and merges neighboring local maxima closer than the size of the dilation. Locations where the original image is equal to the dilated image are returned as local maxima.

everywhere that the image_cube has a local maxima, except in the regions marked for exclusion in the exclusion_cube. This operation is expressed as the bitwise AND of the local maxima AND the inverse of the exclusion layer, AND the image cube values greater than the low_threshold.

In other words: Likely maxima are found in the DoG image pyramid, with horizontal and vertical directional dependence, with the understanding that only a real trace feature is likely to be found at successive levels of blur. In order to limit these maxima, we make sure that features that are part of the laplacian or sobel edges, and that are part of the dark background (as determined previously) are not included. Next, we will combine the information found in all these levels of the image pyramid to make a final determination of what constitutes a real image feature.

We now look for those values that we call ridges, which represent those features most likely to be representing the centerline of a trace. To find these ridges, we collapse the image cubes into a single 2D array by selecting only those pixels from the local maxima that are true across all levels of the image cube. The result is a 2D array of likely ridge lines in both the horizontal and vertical directions. We further limit the ridges to those values that are greater than the high_threshold parameter, noting that this threshold is applied to the image at a point in the process where a pixel has an 8 bit value and therefore can be thresholded. Finally vertical ridges are limited to those that are greater than the threshold OR that satisfy a minimum length and connectivity. These ridges are saved and later will be associated with actual segments in the image.

Results for horizontal ridges are shown in Figure 16.

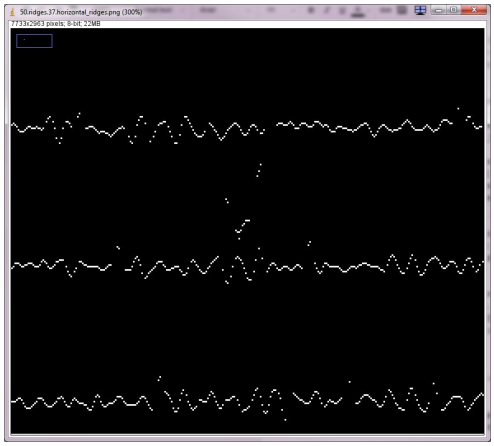


Fig. 16. Horizontal ridges

And the vertical ridges are shown in Fig. 17.

Fig. 17. Vertical ridges.

The ridges are limited to single pixel high features that will later be used to define the trace centerlines. At this point, centerline data is not necessarily continuous, reflecting the conservative nature of this ridge finding algorithm. Experience has shown us that as the algorithms are currently configured finding more centerlines has the unwanted outcome of also pulling in more noise. This noise requires significant editing, which opposes our goal of creating an automated process capable of processing very large numbers of seismograms. Future work will improve the ridge detection algorithm to be less conservative in its selection while improving accuracy.

Segmentation

The above steps have flattened the background, and determined centerlines of traces. We now need to perform a segmentation operation, which identifies traces as features, and separates them from non-trace background features. This module returns 'image_bin', a 2D boolean numpy array in which foreground pixels are True. Background pixels are those that are not part of a trace and are returned as False.

The algorithm accomplishes this by performing a watershed algorithm to segment the image into foreground (traces) and background features. It uses the watershed algorithm skimage.morphology.watershed[14]. The algorithm requires an input image, and seed regions from which the flooding will propagate until 'basins attributed to different markers meet on watershed lines.' These watershed lines then represent boundaries between background and foreground.

The module begins by finding pixels that definitely belong in the background region; these will be the background seeds of the watershed algorithm. The input is a grayscale image (a 2D numpy array) and returns a 2D boolean numpy array where the seeds of the background region are True. One set of pixels used to determine background seeds are the is dark_pixels as defined previously determined in the flattening module.

[14] http://scikit-image.org/docs/dev/auto_examples/plot_watershed.html

The other set of background seed pixels is found by performing a morphological erosion on the gray scale image to remove foreground objects. The boolean image 'local_min' is returned from this image. A bitwise OR operation is performed on the dark_pixels OR the local_min. This returns the boolean array 'markers_background'. Anything that is TRUE in this file is background. These are the background seeds for the watershed algorithm.

To find the foreground seeds, the ridge finding algorithm once again runs find_ridges, but this time the input images are the input grayscale image and the boolean of background pixels. The return logical, called markers_trace, is a bitwise OR of the horizontal ridges OR the vertical ridges, i.e. all the ridges. These are the seeds for the foreground region.

The watershed algorithm is then run on the grayscale image with the foreground and background seeds.

Small objects, both light and dark, are removed using min_trace_size=6, min_edge_length =4, and a connectivity =2. These parameters can be further studied in future work.

The output is a matrix with segmented regions identified with unique labels, representing each trace features. These label values are used in subsequent operations.

Skeletonize

The previous segmentation procedure provides objects that are considered part of the foreground, i.e. objects that are positively identified as traces. This information is used to find the image skeleton.

The medial axis transform is used to find skeleton, using the scikit-image transform[15]. The image skeleton is a one pixel wide object representing those points in the object that have more than one (and almost always exactly two) closest points on the image boundary. While this could be interpreted as the actual centerline of the trace, it does not necessarily coincide with the previously determined ridges, the latter of which represent maximum intensity or minimum second derivative points of the trace's intensity profile.

We use the skeleton in the subsequent step to identify intersecting traces.

A typical output of the skeletonization process is shown in Figure 18.. Note the existence of numerous 'spurs' off the main skeleton. The 'Intersections' algorithm in the next

[15] http://scikit-image.org/docs/dev/auto_examples/plot_medial_transform.html

step eliminates these spurs with a pruning algorithm.

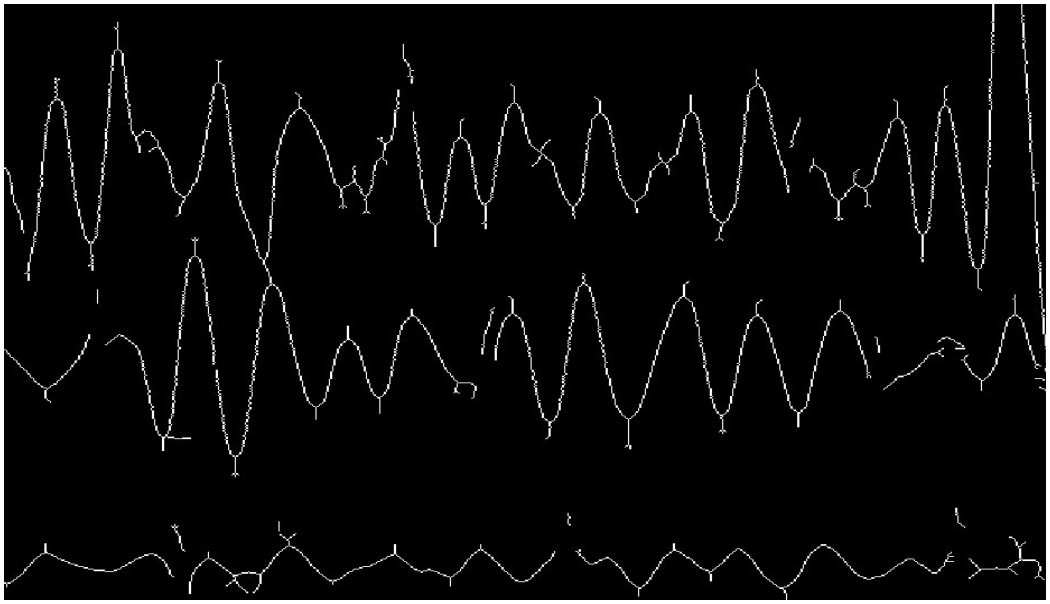


Fig. 18 Skeletonized image.

Intersections

When traces cross during seismic activity, it is important to identify these intersections for several reasons. Of greatest importance is the need to break a trace at an intersection so that it isn't joined to another, different trace. Also important is the inability of intensity methods to determine the centerline of a trace when overlapping intensity profiles of two traces would lead to false centerline data. Therefore, we locate intersections and eliminate the data within the intersections in our final centerline data. While this leads to gaps in the data, future work will develop algorithms to fill the gaps once the proper path has been determined by the later-described connection algorithms. Our philosophy is to be conservative on feature identification in order to minimize false data that can be laborious to remove manually.

Intersections are found and the program saves the collection of intersections to a file as a GeoJSON Feature Collection. Each intersection is saved as a Point with a "radius" property. It also saves the collection of intersections to a file as a black and white image, where white pixels represent intersections. The collection of this and other debug images allows us to troubleshoot modify the pipeline accordingly.

Because skeletonization leads to many false intersections, which we refer to as 'spurs,' the image skeleton needs to be pruned[16]. Our pruning algorithm identifies and deletes dead ends and false intersections.

A dead end can either be a true end to a trace, or it can be a spur off of the trace (and it can also be an intersecting trace, but for now we will just talk about spurs). Therefore a true dead end needs to be differentiated from a spur. To differentiate between spurs and traces

[16] Many pruning algorithms exist. A simple reference is found at https://en.wikipedia.org/wiki/Pruning_(morphology).

that happen to reach a dead end, the above mentioned dendrite threshold represents the minimum ratio of the displacement of the pixel path with respect the width of the trace where the pixel path connects with other paths for the path to be considered a trace. If the ratio falls below this threshold, the path is considered a "spur" and removed. This method is relatively effective at removing spurs while preserving true ends of traces.

Once the pruning is done, true intersections are found. This is done by finding pixels in the skeleton (labeled True in the image) that are adjacent to three or more other pixels in the skeleton. Adjacent, in this case, includes the four diagonal pixels. The size of the intersection/junction is found by getting the distance for the shortest paths from each junction to the edges of the trace, using the previously calculated distance transform of the image. The distance transform returns the distance from an intersection center to the closest edge point of the segment in which it resides. This tends to overstate the size of the intersection (though not by much), but assures that trace centerline data is minimally confounded by adjacent, intersecting traces.

Trace assignment

Nearest meanline assignment

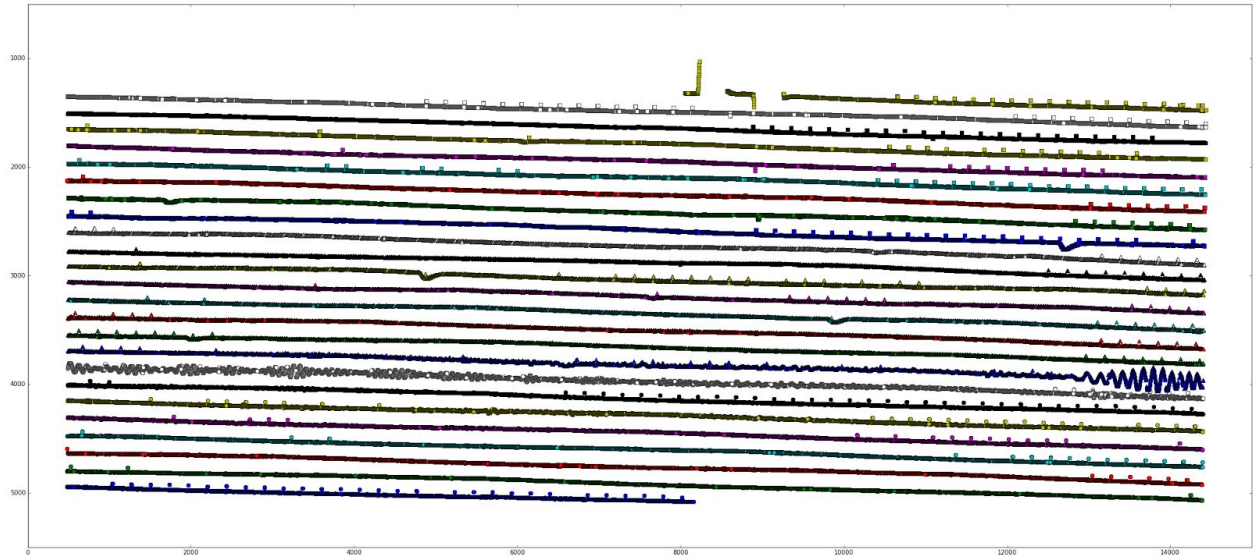
This version of SKATE uses only a single segment assignment algorithm. We have investigated other connection algorithms and will describe them later in this section, and anticipate including them in future work to greatly improve the accuracy.

For now, the algorithm uses geometric methods that takes segment (not centerline) data as inputs, and extracts segment endpoints (useful for future MTSP algorithms), and the average Y value and standard deviation of Y values for each segment.

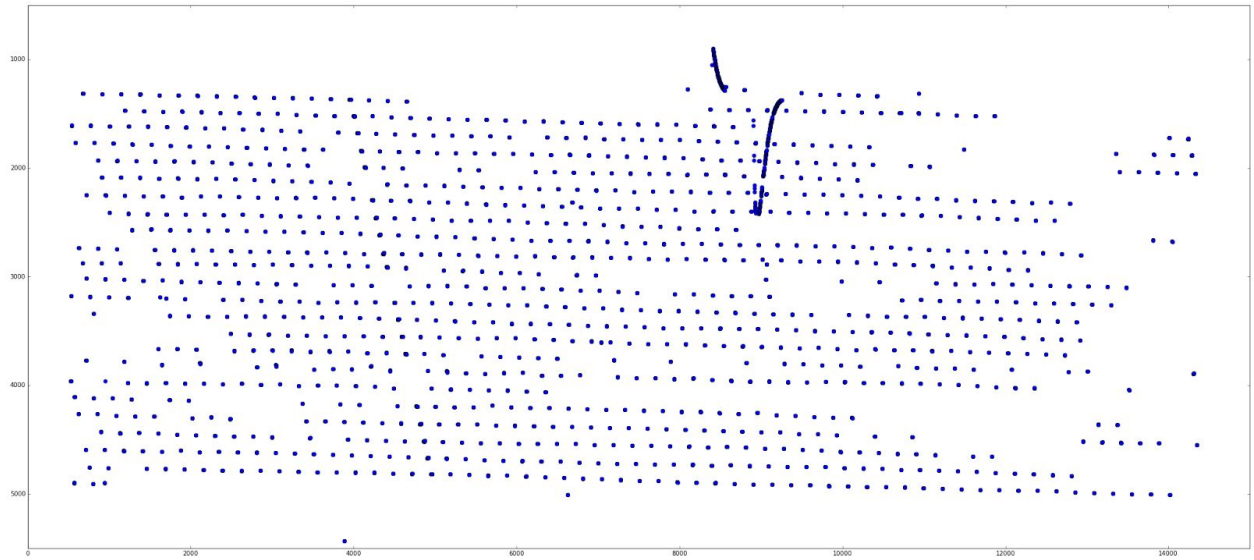
The algorithm uses the average Y-values of all segments to assign segments to the nearest meanline for use in a connection algorithm. It builds a database of all of the meanlines, with information on their slope, which segments have been ‘assigned’ to which meanlines, the distance in y that the average y value of the segment is from the point on the meanline in the middle of the segment’s x domain, and the x domain that each segment assigned spans. With this information, initial assignment of segments to meanlines is done by assigning them to meanlines that are within 45 pixels of the segment’s average y-value.

Segments that were unassigned to any meanline are put into a list of orphaned segments. One by one, the orphan segment’s nearest neighboring segments in the x-direction that have already been assigned to meanlines are examined. If the orphaned segment is next to a segment whose meanline is missing the domain of that orphaned segment, the orphaned segment will be added to that meanline. In this way, orphaned segments are assigned until there are no remaining orphans with neighbors whose meanlines are missing the domain of the orphan.

Segments are assigned to a meanline based on the distance of the average y of the segment to that meanline. In this example, the maximum distance away is 45 pixels. Some timing marks were also captured this way. Different colors represent different meanline assignment.

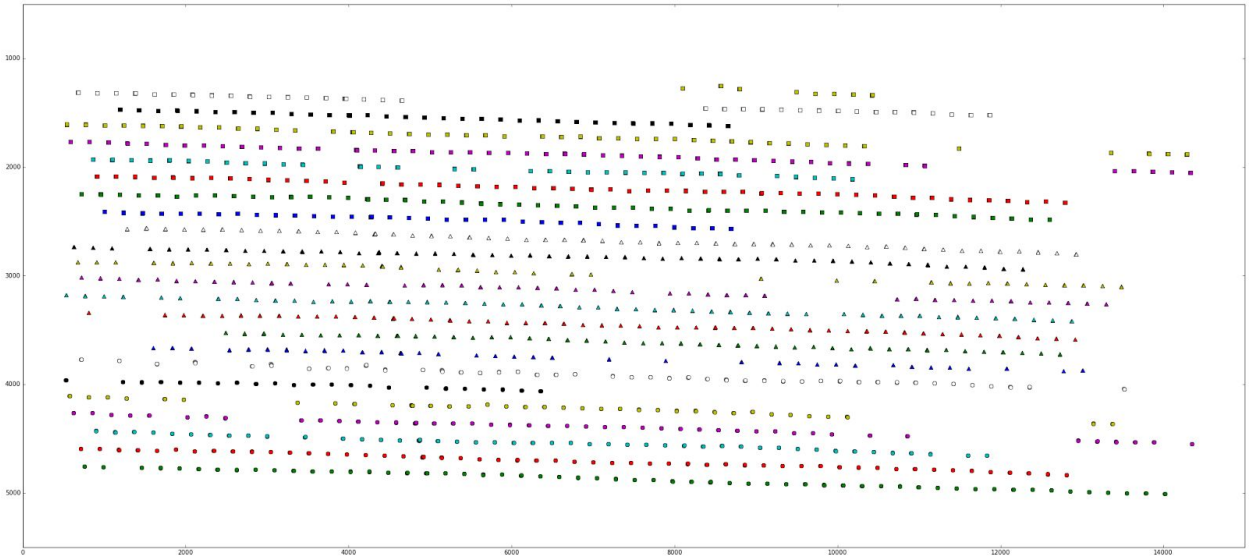


Unassigned segments are shown below. They are all blue. Most are timing marks.

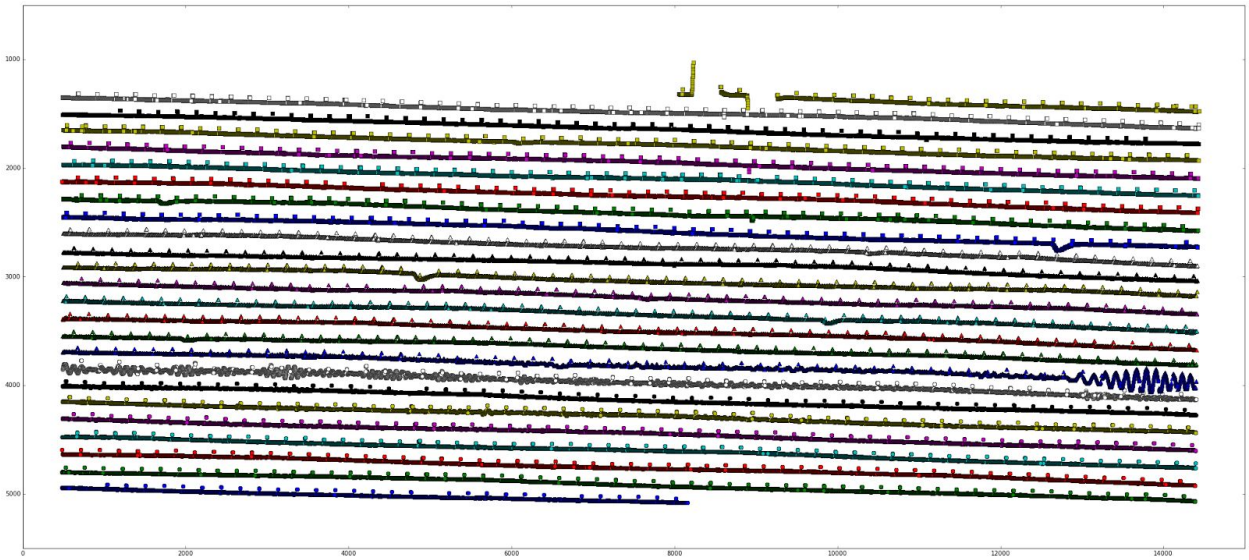


Timing marks are extracted using a size discriminator, with a maximum search distance of 100 pixels above a meanline. These potential timing marks are then passed through a distance discriminator, which keeps only those features with a distance between them of 232 pixels (plus or minus a small error value). This pixel value is based on the measured mean distance on a full size WWSSN long period image with 25 meanlines; for our archive, this corresponds to a scanning resolution of 3.87 pixels/second. Once they are positively identified

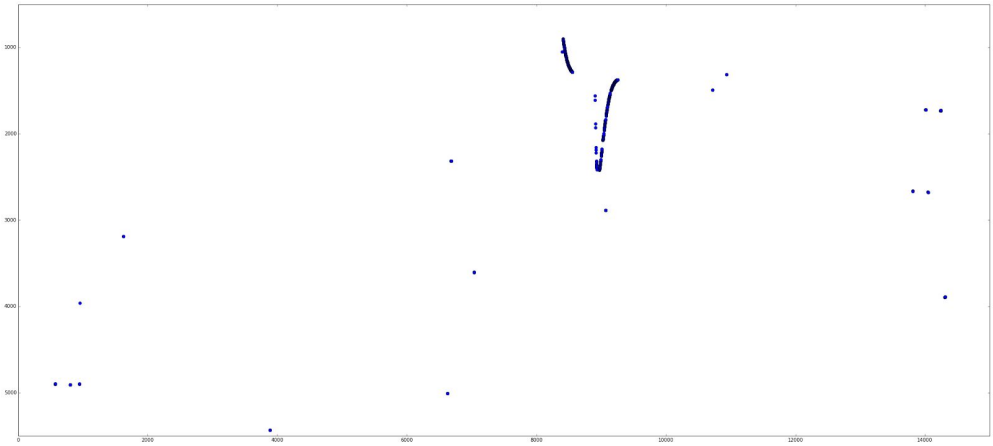
as timing marks, they are assigned to their nearest meanline in the +y direction.



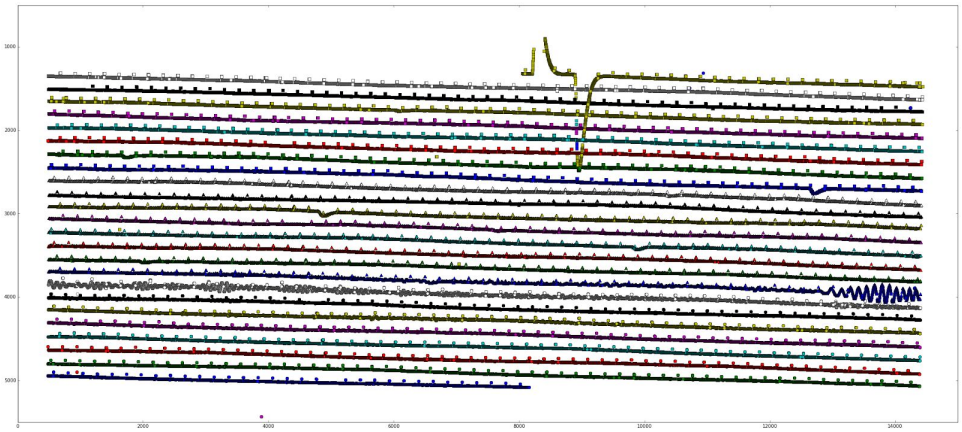
The timing marks are then overlaid on the meanline assignment plot:



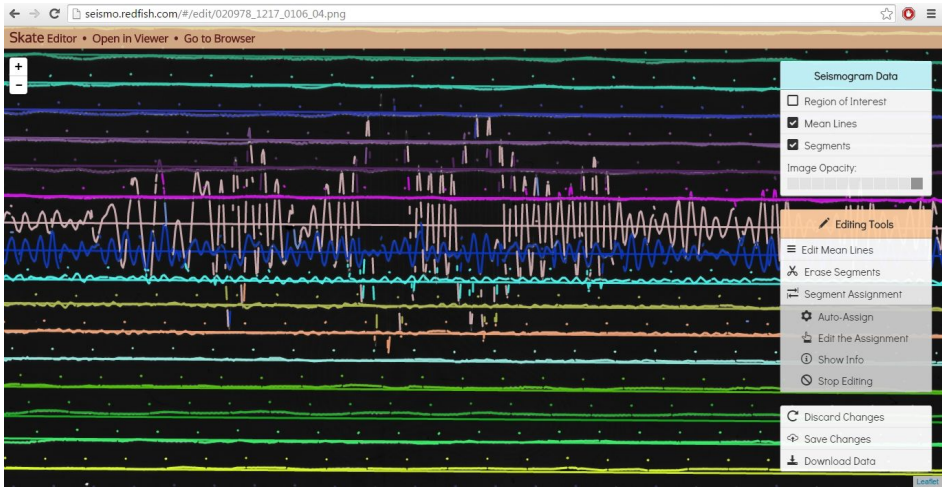
And the plot of unassigned (orphaned) segments shrinks to the remaining segments.



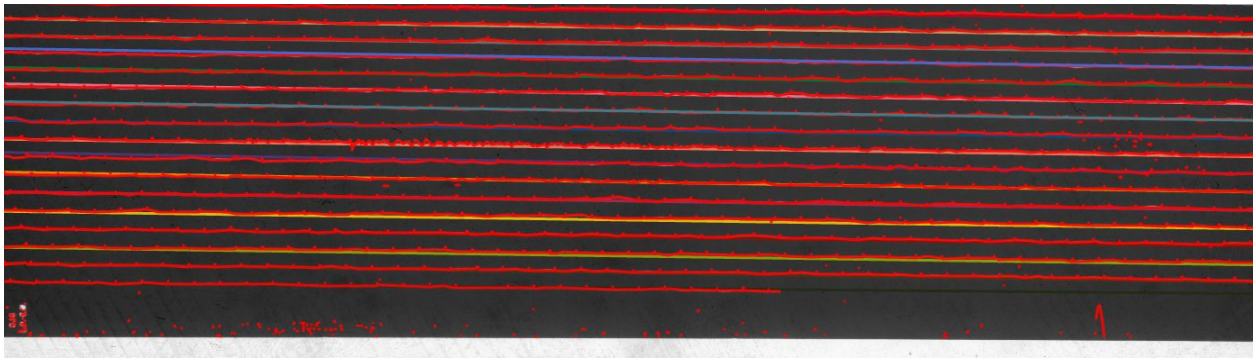
These remaining segments are assigned to the closest meanline who has not been assigned segments that share the domain of the remaining segments.



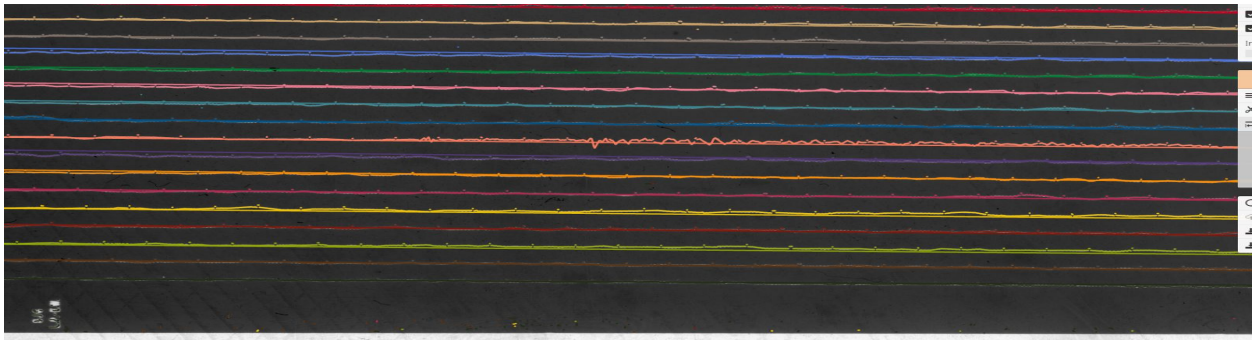
Some more results of a more active seismogram are shown below, showing decent assignment, still using only the nearest mean-y algorithm, with this image shown before any additional editing was performed. Adding more decision tools will continue to improve this process.



Even this initial geometric assignment algorithm eliminates noise features in the course of the process, by eliminating unassignable features. Compare the two figures below that show before and after segment assignment, without any editing done on the image. In particular, much of the noise at the bottom is eliminated. This ability to eliminate features that “don’t fit” will be significantly developed in IIB work.



Before segment assignment.



After segment assignment, with no additional editing.

Other Segment Assignment Methods

We have investigated other trace connection and assignment algorithms, and will only list the most successful methods here. They are discussed in detail in our Phase sequential IIB application.

The segment assignment algorithms are:

- Clustering methods based on Frey-Dueck and k-means techniques.
- Multiple Traveling Salesmen using agent based methods to find the most likely paths across each hourly portion of the image.
- Bayesian methods with (at least) Kalman filters to identify the most probable connections for segments.

These methods will be combined in a multi-criteria decision making matrix to complete a trace path across each hourly portion of the seismogram. They also have been shown to be very useful in eliminating noise and other spurious features.

Editing

A detailed discussion of editing the processed seismogram is covered in the User Manual. For convenience it is included at the end of this report. In summary, the user can take any processed image and edit the meanlines and segments prior to running the connection algorithm. The steps in editing are shown in the flowchart of Figure 19.

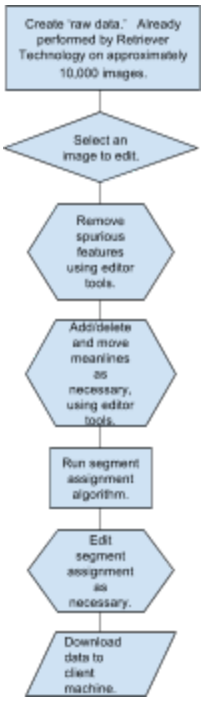


Fig. 19 Flow chart of the editing process.

UI and computer architecture

As we discussed at the beginning of this report, this Phase II effort focused on developing and deploying a web-based platform that created a framework for our interdisciplinary efforts as well as the foundation of our final software package. It is both a technical (algorithms are deployed) as well as a management (all efforts are shared and documented, using a variety of tools including Github to create and monitor efforts) tool that allows for development, oversight, and interactive project management.

We have uploaded our entire repository of scanned WWSSN seismographs - approximately 154,000 high resolution .png images - and processed over 30,000 of these using Amazon S3 and EC2 cloud computing resources. This database of seismograms is fully searchable by date, type and geographical location. The results are editable, and the completed timeseries data is downloadable both in JSON and .csv formats. For all users, we have made all unprocessed seismograms available for download at no charge.

We have created a user interface, available at seismo.redfish.com, that allows users to access all the scanned and processed seismograms, including all editing and downloading of time series features. This section of this report will outline the features of the UI.

The UI

The user interface is a service-oriented web-based system that has been deployed on a host server using Amazon Web Services (AWS). There are numerous reasons supporting this choice of architecture, including cost, reliability, ease of use, and scalability.

For end users, a web-based UI greatly simplifies software operation. There is no software to install; instead, the user only has to navigate to seismo.redfish.com to begin using the software. For Retriever Technology, the creator of the software, it greatly simplifies software operations to host it on AWS. If, for example, we were to attempt to host the remote site on our own server, we would be required to set up, maintain, backup and update a sophisticated web server architecture, with expert tech support when machines fail.

Digitizing the image, that is, extracting centerline data and outputting time series data, is the only compute cost going forward. Our current pipeline, which hasn't been optimized for speed or memory usage, nevertheless costs only $0.02 per full sized WWSSN image to digitize. Storage costs for the current set of scanned WWSSN images amount to about $1500 per year, but this cost could be eliminated if an appropriate agency, such as IRIS[17] or the USGS chooses to add our archive to an

[17] Incorporated Research Institutions for Seismology. http://www.iris.edu/

online repository elsewhere. Because the costs of digitizing have essentially been front loaded by this Phase II SBIR, follow on costs are nearly negligible.

There are many benefits to our architecture. It is available from any computer, anywhere, with the only requirement that of a relatively modern browser. Computer CPU and RAM requirements are very lax, as all compute intensive operations run remotely on our AWS servers. The editing tools run in the user's browser, and are not compute intensive. There are no packages or licenses required from the users which would otherwise incur additional costs, and in many cases require specialized software knowledge. For example, there exist numerous toolboxes written in Matlab, and these require an upfront investment of thousands of dollars. If additional compute power needs to be added, the AWS servers are easily scaled up (and have been during the course of this project when we processed 30,000 images. After the processing, we turned off the servers and stopped paying for unneeded instances.) Extreme instances of over 120 GB RAM can easily be spun up for unique problems; such compute power is essentially unavailable on any type of consumer computer.

Terminology:

A list of the terminology used in describing our web based system is given below.

- "App" or "browser app" or "application" means the bundle of code that runs in the user's browser. This consists of the file browser, the viewer, and all of the editing tools that the user finds when accessing the application at seismo.redfish.com
- "Server" or "web server" means a piece of code sitting on the remote host machine (seismo.redfish.com). It is up 100% of the time and ready to reply to requests sent to it by the "app". The server can save metadata to S3, query the database for files, verify authorization, etc. It's the muscle behind the browser app.
- As described previously, the"Pipeline" is the image processing system written in python that processes the images and extracts time series information. Currently this is offline in order to minimize costs. However, it can be put online at any time, funding permitting.
- "S3" is a pure storage service by Amazon. Pure storage means that one can't run code on S3; it is just hard drives, not CPU.
- "EC2" is where the web server and pipeline run. It's a "compute service" provided by Amazon. It provides CPU/RAM and can be scaled up and down dynamically.

The pipeline is the previously described set of image processing algorithms that is written in python/numpy/scipy. Python is developed under an OSI-approved open source license, making it freely usable and distributable, even for commercial use[18].

[18] https://www.python.org/about/

The web server is implemented in node.js. Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient[19]. It is scalable, allowing it to handle large volumes of requests that our work might create in future development.

The browser app is written in Javascript/Angular, allowing for open source development that can build dynamic views of data that change immediately in response to user actions. It is well suited for SKATE's browser based implementation.

In addition to the 5 TB of image data stored on Amazon S3, all metadata and thumbnail images are also stored on S3. A Mongo[20] database stores the file and station data. Mongo was chosen because of its speed and simplicity; it is an open-source document database designed for ease of development and scalability. It is well suited for our needs especially with our simple data structures, which in this case consist only of files and stations. Each file object has:

- name (used to locate the actual image on S3)
- station ID
- status (not processed, processed, edited, problematic, etc.)
- date and time of recording
- long/short, up/down, east/west

To render each seismogram in the browser, we use an advanced image tiling technique that makes it unnecessary for the user's computer to deal with opening large images. When viewing a seismogram, we only display the viewable part of the image, sampled down to match the current zoom level. The result is a much smoother experience for the user, who only loads about 700 kilobytes at any one time, whereas the original image is ~30 megabytes. This rendering technique dramatically speeds up browsing and editing activities, and it optimizes data transfer from AWS servers to the client computer. We utilize Leaflet, an open source JavaScript library for interactive maps. It works efficiently across all major desktop and mobile platforms out of the box, taking advantage of HTML5 and CSS3 on modern browsers while being accessible on older ones too[21].

The web server interacts both with the mongo database and with S3. An example of server↔mongo interaction looks like the following:

The user utilizes the search feature in the app to search for "Edited" seismograms. The app sends a search query to the server. The server searches the mongo database for files whose status is "Edited." The server returns a result set of the "Edited" files.

An example of server<->S3 interaction looks like the following:

In the browser app, the user opens an image in the viewer. The viewer requests image tiles from the server. The server grabs the image from S3 and generates the requested tiles from it.

[19] https://nodejs.org/en/
[20] https://www.mongodb.org/
[21] https://github.com/Leaflet/Leaflet/blob/master/README.md

The Web App runs fully on the client side and interacts with the web server. This means when the user navigates to the UI, their web browser downloads the entire app, and the application from then on runs fully inside the browser on the user's computer. The app occasionally requests and pushes data to the server, such as saving edited data, or downloading existing metadata.

A flowchart of the overall process is shown in Figure 20.

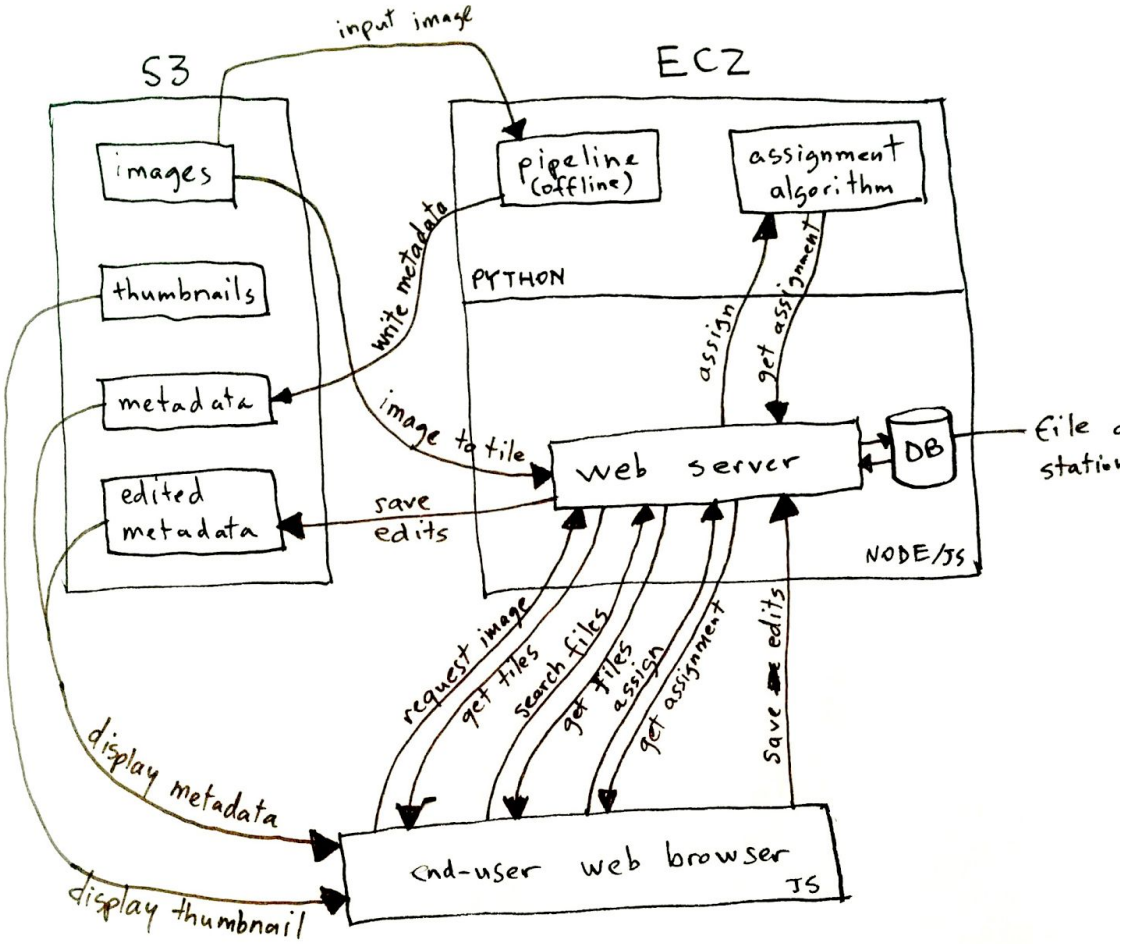


Fig. 20 Flow Chart of the Overall Process

The Pipeline in Python

The pipeline is written in Python, and is also available upon request as a package of modules that can be run on a user's own computer. There are limitations, particularly in the size of the image that can be run. As previously discussed, a 30 GiB EC2 instance is required to process a full sized image. Nevertheless, the desktop version has been very useful for debugging and algorithm development, either by running on a smaller selected area of a full resolution seismogram, or by running on a full sized but reduced resolution image. Over 30 debug images are available, allowing in-depth

analysis of any number of modules. There is no option for editing of the processed seismogram; this function is only available through the UI.

Concluding remarks

We believe that SKATE is the fastest and most accurate digitization program available. With successful IIB funding we will be able to further develop it to become even more automated, to the point that the goal of digitizing millions of analog seismograms will finally be within reach. We know that we can digitize all of these right now; it is only a matter of reducing the cost and increasing the accuracy that will allow us to actually do it. Our path forward will reduce digitization costs by an order of magnitude, allowing for 1,000,000 seismograms to be digitized for only $2,000. The accuracy improvements that are in our IIB proposal will output data that in many cases will require no editing at all, while in other cases the most active seismograms will have a suite of automatic and manually activated tools that will allow for very rapid editing of some of the most challenging images.
We are striving to make SKATE the premier seismogram digitization program.
In order for SKATE to gain acceptance and persistence in use, we will operate under the following philosophy.

- Free or extremely low cost fees. SKATE is hosted and run on Amazon's S3 servers and EC2 compute instances. Users need only direct their browser to our website at seismo.redfish.com to begin using the software. There is no need to purchase or download software, and in particular the high performance computing that EC2 instances offer are significantly more performant than could reasonably be purchased (and maintained) by a typical single end user[22]. Currently, the cost to process a single seismogram up to the point of editing ius $0.02. There is additional cost in storing our archive of over 150,000 WWSSN seismograms, plus monthly costs incurred in keeping the site active. Our IIB proposal will cover these costs for up to ten years at a cost of less than 3% of the total IIB budget. Nevertheless, we will explore and develop paywall access as needed.
- Open source software. Currently, the software is stored on a Github repository with access limited to team members. In IIB and beyond we will open this up to all users for downloading, and implement version control protocols. The software will reside in an organic environment that will allow continuous improvement and customization as the product grows. As an example, users could develop tools to digitize specific seismogram types and correct for concomitant recording distortions beyond the WWSSN seismogram types we are currently focusing on.

[22] The current r3.xlarge instance has 30 GiB of memory and 4 vCPU of processing power. Significantly more powerful instances can be called. See https://aws.amazon.com/ec2/instance-types/

- Web-based. This is a core distinction of SKATE. The software will always be online, and always be available to all users. Access as simple as navigating a URL to a website will insure its increasing growth and popularity.
- Finally, the accuracy and functionality of SKATE is, and will continue to be, unmatched by any other digitization program. Our sophisticated algorithms are unique, and will only continue to improve and add functionality. Digitizing a seismogram in seconds on a web site and pulling out a complete timeseries solution in a very short time is unique, compelling and is the only practical way to unlock the information currently unavailable in millions of seismograms.

Our Phase II work has been successful because it has been collaborative and open source. We will work to create a user community that can network, collaborate, and further develop this software. Successful IIB funding will ensure that this work will persist, will meet non-proliferation national security goals, and will expand the scope of seismology by opening up the vast archive of analog data to modern analysis techniques.

Caltech archives

As discussed, we will develop the pipeline to handle a wide range of image and seismogram types. We note that we have already test run the pipeline on other seismogram types, from other sources. The images below are from the Caltech archives that were scanned by Google as part of the Google Books project.

Meanlines were found without any alteration to the pipeline code and are shown in Figure 21[23]. The colored image shows that segments were also found and identified. Different segments have different colors. The centerline data is not shown, and we did not run the assignment algorithm on this image, but would expect them to work well given the straightforward nature of this image.

[23] Image ID: http://ds.iris.edu/seismo-archives/projects/caltech_archive/1929.5.12%20to%201929.5.2%20SCSN/Page_2_images_2_000172.%23img%23.JPEG

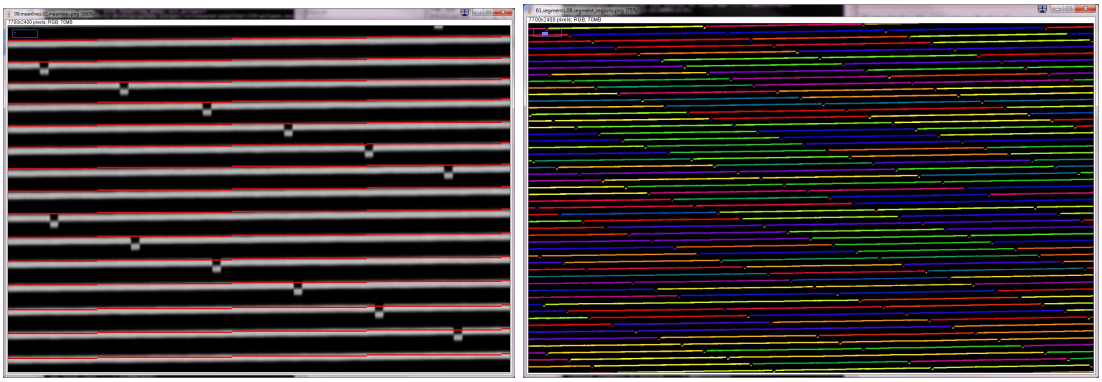


Fig. 21 Meanlines and segments from the Caltech archive.

And here are a few references concerning archives and digitization.
http://srl.geoscienceworld.org/content/86/1/255.full
http://ds.iris.edu/seismo-archives/projects/caltech_archive/Caltech_Seismograms_v2.pdf
http://www.iaspei.org/downloads/IASPEI_SeismoArchives_Project_summary_2011.pdf
http://ds.iris.edu/seismo-archives/projects/

Appendix A

The user manual for SKATE is presented below for reference.

SKATE

USER MANUAL V1.0

Table of Contents for the User Manual

- Appendix A
- USER MANUAL V1.0
  - Introduction
  - Current Status and Available Images
    - Requirements
  - How to use SKATE
  - Operating Instructions
    - Definitions
    - Login
    - Searching
    - Browsing and Viewing
      - Zooming in and out
    - Editing
      - Editing Tools
        - Edit Mean Lines
          - Moving a meanline
          - Deleting a meanline.
          - Adding a meanline
        - Erase Segments
      - Segment Assignment
        - Editing Segment Assignment
      - Saving or discarding changes. Downloading data.
        - Discard Changes
        - Save Changes
      - Download Data
        - roi.json
        - meanlines.json

segments.json
assignment.json
assignment.csv
mSEED
Logout
Final

Introduction

USER NOTE:

SKATE is a work in progress, and the current version reflects research and development that was done up to the completion of current DOE SBIR Phase II funding. It is currently not complete, and users should understand that the purpose of making this software publicly available is to allow qualified researchers to evaluate the work to date, and to make detailed comments and suggestions as to how it can be improved. Known issues include improved meanline detection, significant reduction in spurious features, and most importantly incorporating advanced connection algorithms that we are developing.

We anticipate future funding that will allow us to complete this software, and encourage users to contact Retriever Technology with any and all comments.

SKATE (Seismogram Kit for Automatic Trace Extraction) is a web-based software tool for the digitization of seismic traces in historic analog seismograms. This work was performed under DOE contract DE-SC0008219.

SKATE is designed to address the need to digitize[24] the millions of historic seismograms that are currently unavailable for analysis. While there exist other digitizing programs, they are typically limited by the need for significant user interaction, slow speed, and/or proprietary software requirements. The design of SKATE specifically addresses these limitations by:

- Web-based architecture. To operate SKATE, the user simply needs to direct a browser to the software's web location at seismo.redfish.com. No software needs to be installed

[24] 'Digitize' refers to the creation of (x,y) timeseries data for seismic traces. 'Scanning' refers to the creation of a digital image from an original paper or film record. A scanned image has not yet been digitized.

on the user's machine; modern browser capabilities allow the entire process to be run on a remote server.

- Fast and parallelized processing. Thousands of seismograms have been processed using Amazon EC2 virtual computers, running in parallel. Processing time per seismogram is currently 15 minutes. We anticipate that this will be speeded significantly in future versions.
- Reduced user interaction. We have designed robust and detailed algorithms to extract and separate real trace information from background noise and other spurious features. Segment connection algorithms automatically track traces across crossings when there is significant seismic activity. User interaction is still recommended and required in order to eliminate spurious features, but easy to use tools and intelligent algorithm design reduces this effort significantly.

This is the alpha version of SKATE, and users should recognize that testing of the product is ongoing. Hence, this manual will contain phrases like 'should' and 'likely,' which indicate that while we believe most features will behave as described, uncertainties such as variabilities in browsers and versions and end user machines can affect the behavior. The user should understand that SKATE is constantly being upgraded, and as such features will be added and subtracted without warning. Please contact andy@retrievertech.com with questions and comments about operation, updates and suggestions.

Current Status and Available Images

Retriever Technology has scanned approximately 150,000 WWSSN images under several contracts, including but not limited to USGS Awards G09PC00116, G09PD02064, and G10PD02236. These images were saved as high resolution, uncompressed .tif files, and were delivered to the USGS. Retriever Technology has used these images to develop SKATE, and has stored them for this project as lossless .png files on an Amazon S3 server. As of now, we have completed two types of digitizing of WWSSN images:

- What we call 'raw data,' meaning that we have processed approximately 10,000 randomly selected images up to the point of automatic meanline assignment, and segment and centerline identification. These data have not been run through the final segment assignment algorithm. This assignment algorithm can be run at any time, but we feel that absent some user interaction and editing, the final result will have less than ideal results. These images were limited to WWSSN long period seismograms from 1970 and on.
- The second type of processed images are listed as 'has edited data.' This is a small subset of approximately 20 images that we have hand edited to remove spurious features and to ensure that the meanlines are properly placed. They were then run through the segment assignment algorithm and have data that is ready to be downloaded. Please refer to later sections in this manual that discuss editing and downloading of data.

- There is a third category available through the UI called ‘no data.’ There are over 150,000 of these high resolution scans available for viewing only. Budget limitations due to Amazon’s EC2 operational costs prevent these from being processed or downloaded at this time.

Currently, users cannot upload and process their own images, nor create ‘raw data’ on any of the ‘no data’ seismograms. This feature will be added shortly, please refer to the website for updates and information on upgrades. What users can do is edit any of the ‘raw data’ seismograms, and run the assignment editor on them. Please note that there are severely limited funds to maintain and operate the web server. As such, we ask that users design their experiments to maximize results while minimizing computational time. Downloading data incurs costs as well, so only download data when editing and all processing is complete.

While SKATE has been optimized for project development purposes for long period WWSSN images, it can readily be configured to digitize short period images, of which there is an example in the ‘edited data.’ In addition, we have successfully digitized data from the Caltech archives, see the following reference for information on this archive: http://ds.iris.edu/seismo-archives/projects/caltech_archive/Caltech_Seismograms_v2.pdf. We are working to expand the types of seismograms that can be digitized. This includes partial images that contain only an event of interest. This effort is limited only by the need to understand and address particulars of resolution, size and other general features found in these images. SKATE’s algorithms are very general for finding seismic features as lines traces. We have even digitized historic mareograms (tide gauge charts).

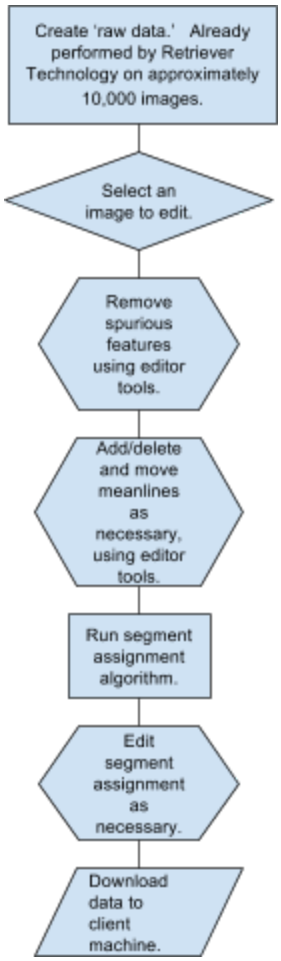
Requirements

The SKATE seismogram processing package is hosted on a remote web server. There is no need to install any software. Computer and browser requirements are:

- Browser. SKATE has been tested in Chrome only. The most current version tested is 45.0.2454.101 m. It is recommended that the user update Chrome to the latest version. There are no known settings or permissions required. Internet Explorer 11 has not been tested but should work. Mozilla Firefox version 45.0.2454.101 m has not been tested extensively, but should work as well. It is highly recommended that Chrome is used. If other browsers are used, it is highly recommended that the user update to the latest version.
- System requirements. Because SKATE runs in the browser, with all editing and processing done on a remote server, client system requirements are light. Reasonable and standard RAM and CPU performance should suffice. Users might want to consider an upgraded graphics card if the display does not respond quickly to commands. A desktop or laptop computer is recommended. Though we have tested SKATE on Android tablets and phones, these devices have not yet been fully optimized.

How to use SKATE

This version of SKATE has been designed to allow users to explore and evaluate its current capabilities by viewing, editing and downloading seismogram timeseries data that have been solved up to the point of having the above described ‘raw data.’ The flowchart outlining the process is shown below.



Operating Instructions

When referring to a command or other information that appears on the UI, those words will appear quoted in italics. When referring to a keyboard action, the actual keystroke will appear in angle brackets, e.g. <+> indicates use of the plus sign on the keyboard. Do not include the angle brackets in the action.

Definitions

Terms used in this manual are defined here.

meanlines. This is the zero-energy line that a trace would follow if there was no seismic activity. Meanlines are the image-wide line segments that we calculate for this zero-energy path, and which are used for subsequent trace assignment and connection algorithms

trace. In a WWSSN image, this is the image path that the incident light beam records on the recording drum's photographic paper. It is the seismic data from which we extract timeseries data.

segments. A seismogram image file consists of foreground and background. Foreground features are the traces themselves. Our task is to identify and separate them from the background. Background includes all non trace features such as noise, and additional features such as hand written notes, etc. Because of timing marks, trace crossings, and the multiple hourly lines found in a WWSSN seismogram, the traces are broken up into pieces. We call these pieces of a seismic trace a segment.

centerlines. Centerlines represent the actual path that a trace is following. It is a single pixel wide object that tracks the middle of a segment.

ROI. The ROI is the Region of Interest in the seismogram. It is the portion of the image that contains the actual seismic traces. In the image below, the ROI is shaded in light blue.

scanning. Seismograms that are scanned are simple image files, usually .tif or .png.

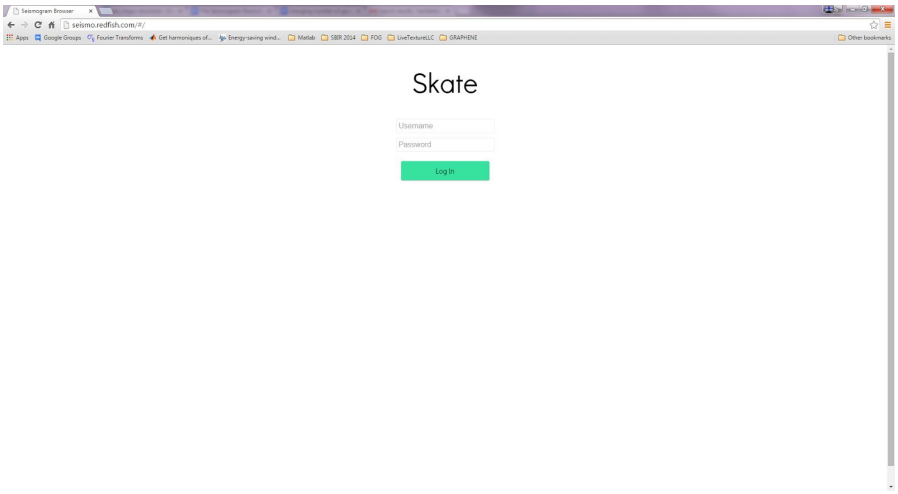
digitize. Digitizing a seismogram takes the scanned image file and extracts the time series that represents the seismic trace's amplitude as a function of time. We output the time series as either a .json file or a .csv. Other formats such as mSEED will be created in future versions.

S3. Amazon S3 (Simple Storage Service) is an online file storage web service offered by Amazon Web Services. Amazon S3 provides storage through web services interfaces (REST,

SOAP, and BitTorrent)[25]. Retriever Technology stores all of its scanned seismograms on S3. In addition, the UI is hosted on S3. Monthly costs accrue for this service.

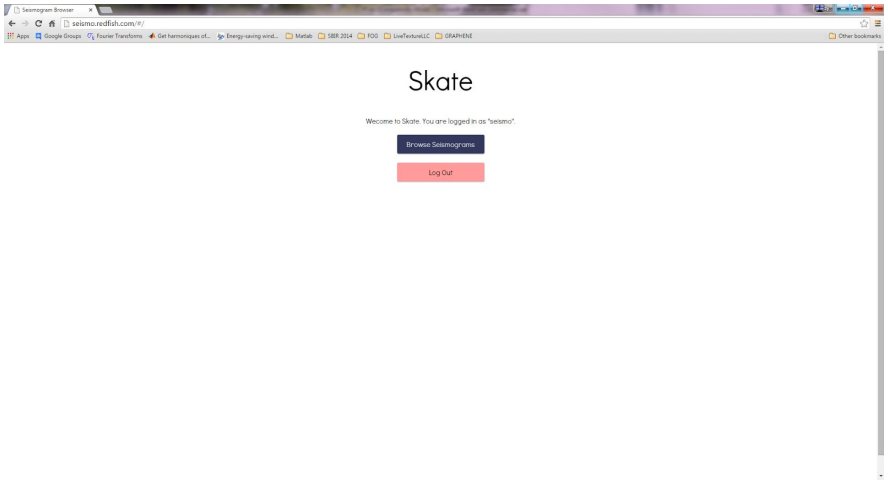
Login

Navigate to seismo.redfish.com The following welcome page appears. Login with:
Username:xxxxx
Password:xxxxx
OR, contact Leslie Casey at Leslie.Casey@nnsa.doe.gov in order to obtain login credentials.
NOTE: Login is no longer required to process, edit and download a timeseries. Also, open downloading of any full sized image in our archive is also now available. Login credentials are only required if the user desires to alter the metadata stored on S3.

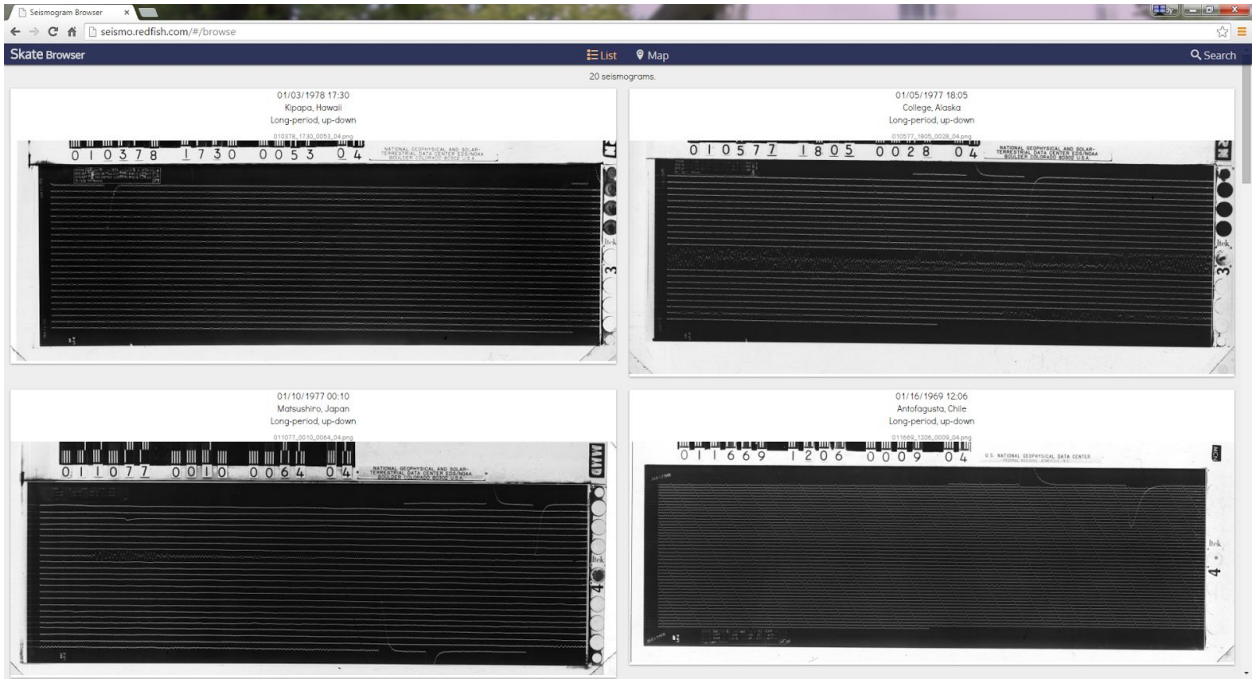


After successfully logging in the following screen will appear. Select '*Browse Seismograms*.'

[25] https://en.wikipedia.org/wiki/Amazon_S3

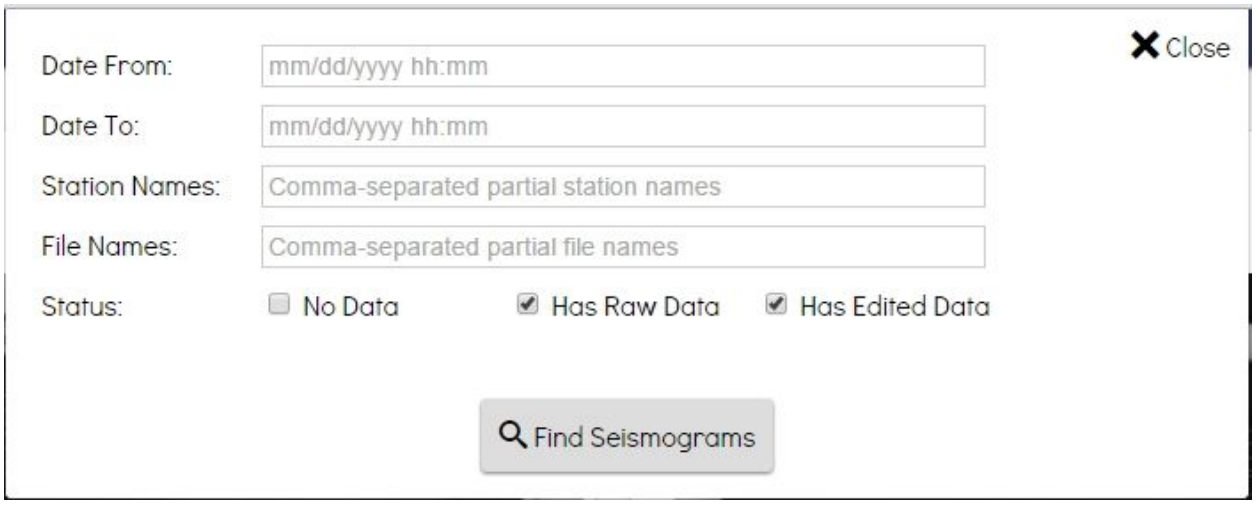


At this point a thumbnail list of seismograms will likely be displayed:



Searching

To search for a particular seismogram, click on the "*Search*" link. The following menu item will appear.



Dates. The Search window allows specific dates and times to be searched. A complete date can be entered. Alternatively, the four digit year can be inputted in the '*Date From*' and '*Date To*' fields. Searching by only putting in a month or a day or a time will not work.

Station names. This allows for searching of one or more stations by name. Type in the three character station name, or a portion of the name, or the name or portion of the actual location (i.e. city and country name). Multiple stations can be searched by separating station names by a comma. Do not search by station number, as this option is not available. Quotes and wildcard operators do not work.
Station names and other WWSSN information can be found in the WWSSN User's Guide at http://pubs.usgs.gov/of/2014/1218/pdf/ofr2014-1218.pdf.

No Data. Searching can include/be limited to seismograms with *'No Data*.' This indicates a seismogram that is accessible from the UI but has not been processed in any way. There are currently 154,658 of these seismogram which represent Retriever Technology's database on S3. Users may search and view these images, but due to cost constraints tied to Amazon's data download fees and EC2 operational fees, these are not available for download or processing. Contact Retriever Technology if there is a particular seismogram of interest.

Has Raw Data. Searching can include/be limited to seismograms with '*Raw Data*.' This indicates seismograms that have been processed up to the point where they have: ROI, meanline, segment and centerline information. They do not have segment assignment information. There are currently over 11,000 of these file, with more being processed on 40 EC2 instances. There will be approximately 50,000 available shortly.

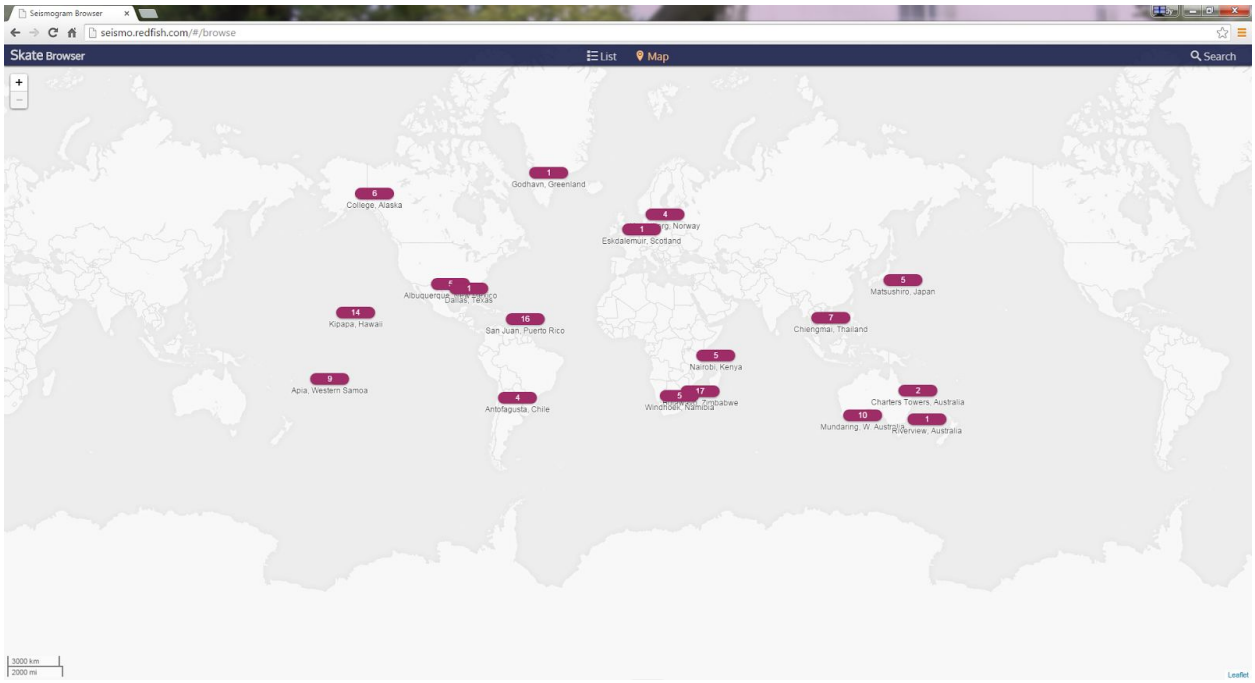
Has Edited Data. Searching can include/be limited to seismograms with '*Has Edited Data*.' These seismograms have the same information as in '*Has Raw Data*', and in addition have been edited to improve meanlines and remove spurious segments as necessary, and then have

been run through the assignment editor. Segment assignment means that segments are sequentially attached to previous and subsequent segments. In other words, the seismogram has been solved.

This is a small subset of images that have been hand-edited by summer interns at Retriever Technology. They are likely very good, though not perfect, in their editing.

Browsing and Viewing

Once a search is complete, a thumbnail list of found seismograms will be displayed. The default "*List*" display is active and will display 20 thumbnails per page. Scrolling down will continue to pull up more thumbnails. Alternatively, clicking on the *"Map*" icon will display a world map with location and counts by station of the searched-for seismograms.



To view seismograms from a particular station, click on the location. The available seismograms will display in the "*List*" view.

Once the seismogram of interest is located, click on the thumbnail to display the image. Navigating the image is done as follows.

Zooming in and out

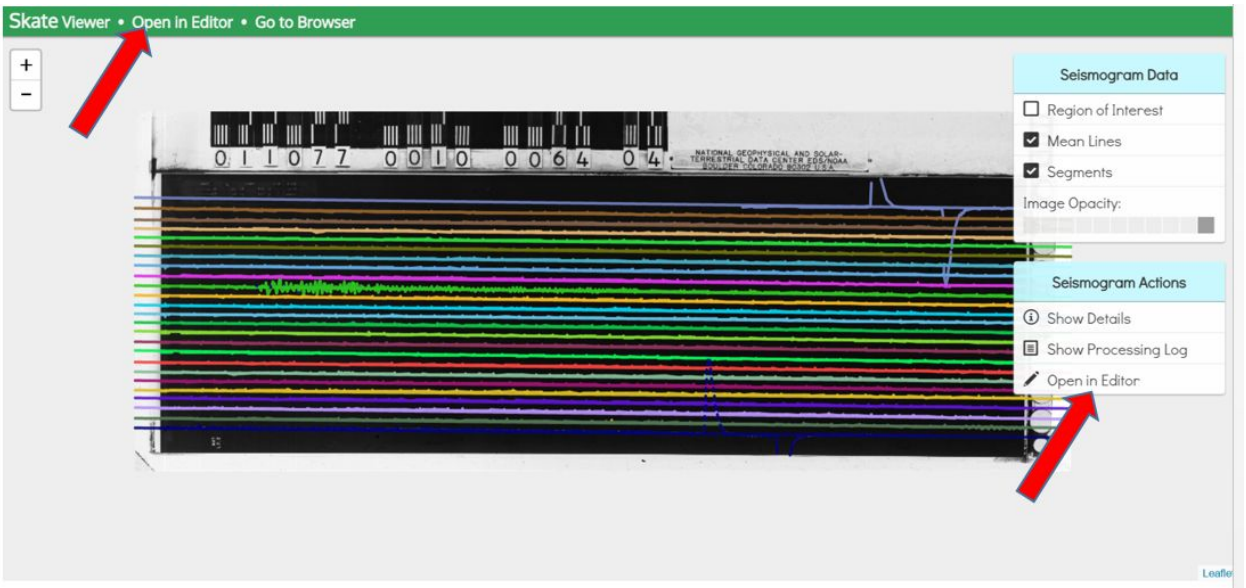
Using the scroll wheel on a mouse zooms in and out on the displayed image. Also, the "+/-" button on the screen allows zooming.

Absent a scroll wheel, the <= > keyboard key zooms in. The <- > keyboard key zooms out. If the "*Seismogram Data*" or "*Seismogram Actions*" menus have been used, it might be necessary to first place the mouse over the image and click once in order for the keyboard zoom function to become active.

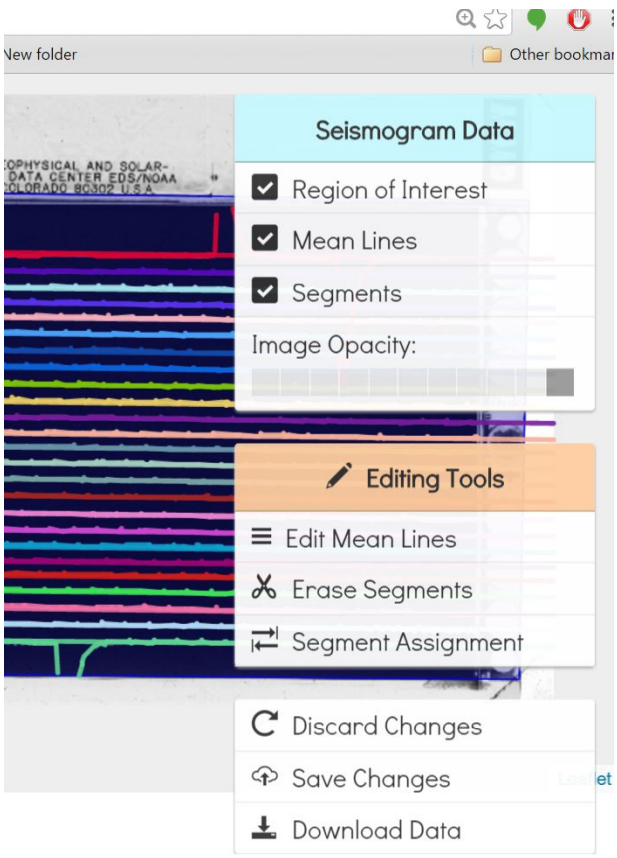
Panning. Left click and hold while dragging the mouse. When zoomed in, the up/down, left/right arrows on the keyboard also should allow panning.

Editing

Editing can be done both before and after segment assignment. To switch to editing mode, click on either of the links shown by red arrows below.



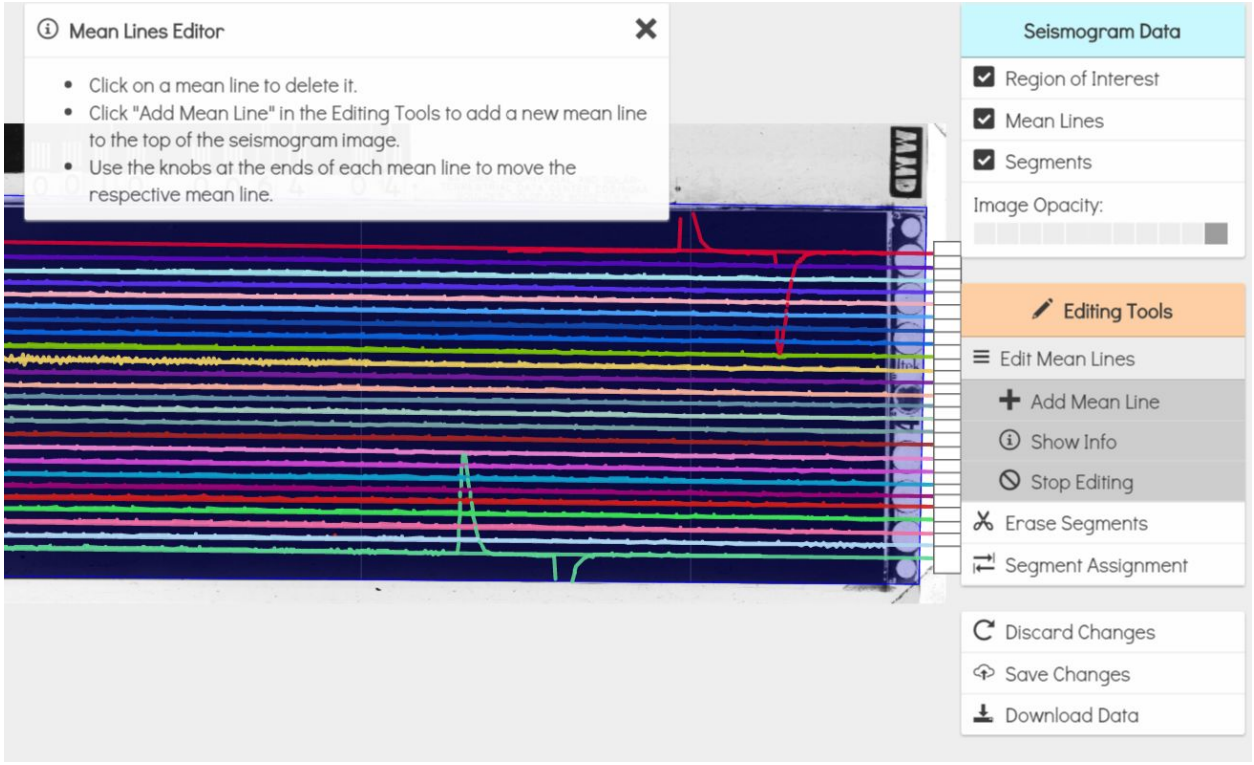
Once in Editor mode, the following screen with menu items becomes available.



Editing Tools

Edit Mean Lines

Clicking on "*Edit Mean Lines*" pulls up the following menus. It might be necessary to click on "*Show Info*" in order to view the "*Mean Lines Editor*" menu box:

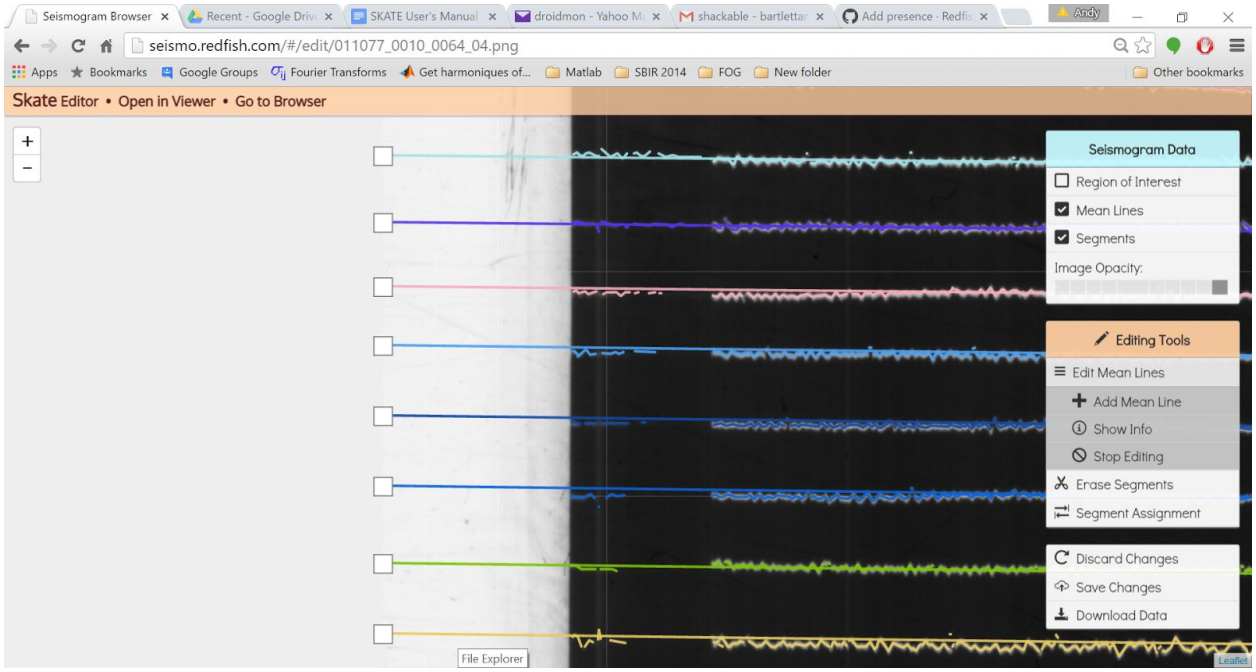


Meanlines are the zero-energy line of the seismogram, and are used for Segment Assignment. However, they are not used as data normalization lines, so it is not critical to exactly align them to the zero-energy line.[26] Images that are listed as "*Has Raw Data*" are those in which meanlines have been automatically calculated. The algorithm is not 100% accurate, so the user should add, delete, or move meanlines in order to cover all lines and partial lines with a meanline.

Moving a meanline

Meanlines have handles on the end of them, and extend beyond the ROI and out to the edge of the entire image in order to make it easy to visualize and manipulate. Zooming in while in the meanline editor looks like this:

[26] Image distortions such as lens distortions, scanning errors, paper shrinkage, etc., make a linear fit to a zero-energy line less than exact. The issue of zero-energy line fit will be handled in future iterations when manipulating the trace information in data space, as opposed to image space.

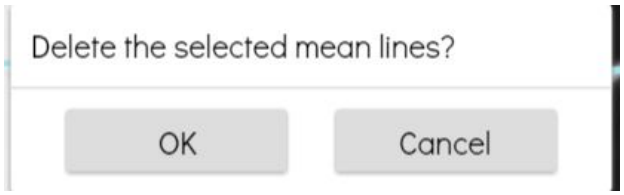


Note that the meanlines in this example are not perfectly aligned with the zero-energy line, but represent positioning that is sufficiently close for subsequent segment assignment.

To reposition a meanline, click on the handle and drag the mouse to the desired position. Repeat for any line that needs to be repositioned. Changing one meanline does not change any others.

Deleting a meanline.

To delete a meanline, click anywhere on the meanline besides its handle. A dialog box will appear:



Select "*OK*" or "*Cancel*" as appropriate.

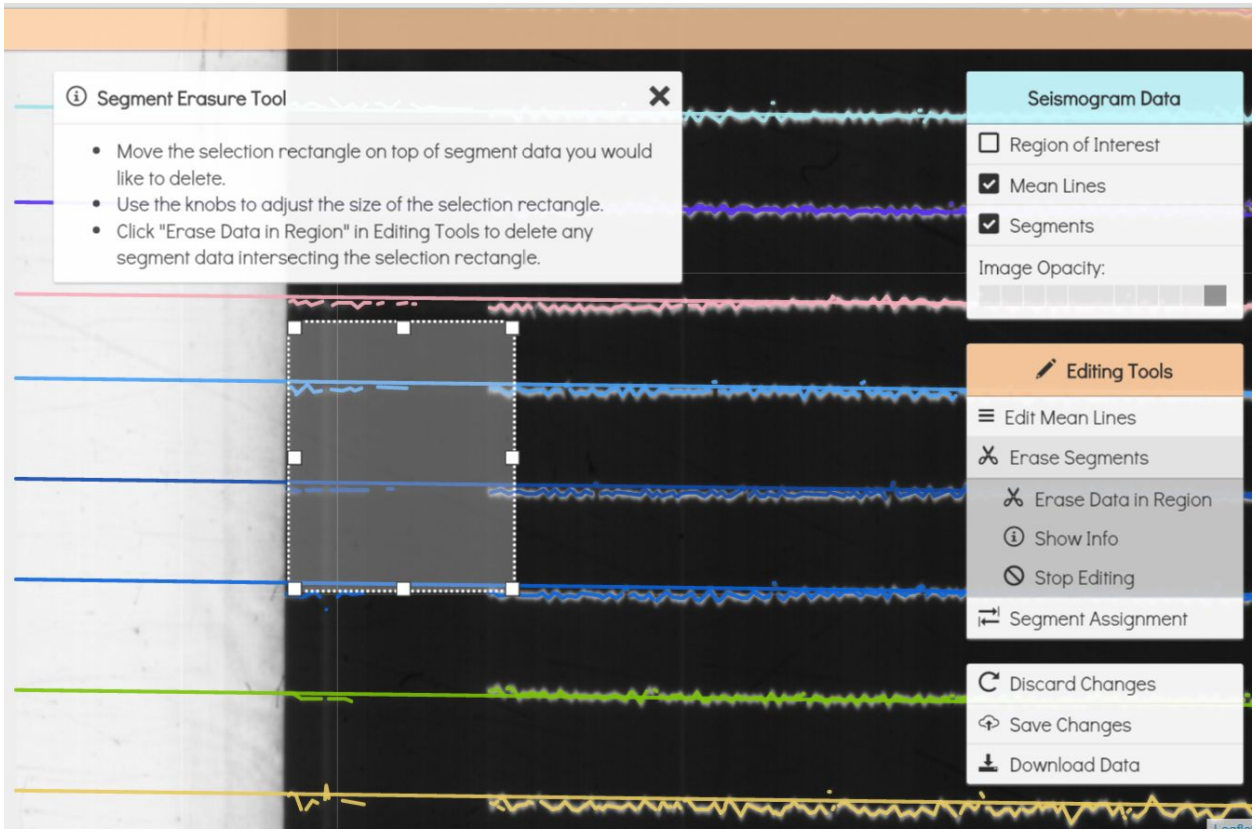
Adding a meanline

To add a meanline, click on "*Add Mean Line*." The new meanline will appear at the top of the image. (Zooming out might be required to see the new meanline.) Multiple meanlines can be added. Note that clicking on "*Add Mean Line*" multiple times will overlay the new meanlines on top of one another. Each needs to be dragged out of the way in order to view the new meanline that is below it. Place meanlines where desired using the procedure to move meanlines described above.

Erase Segments

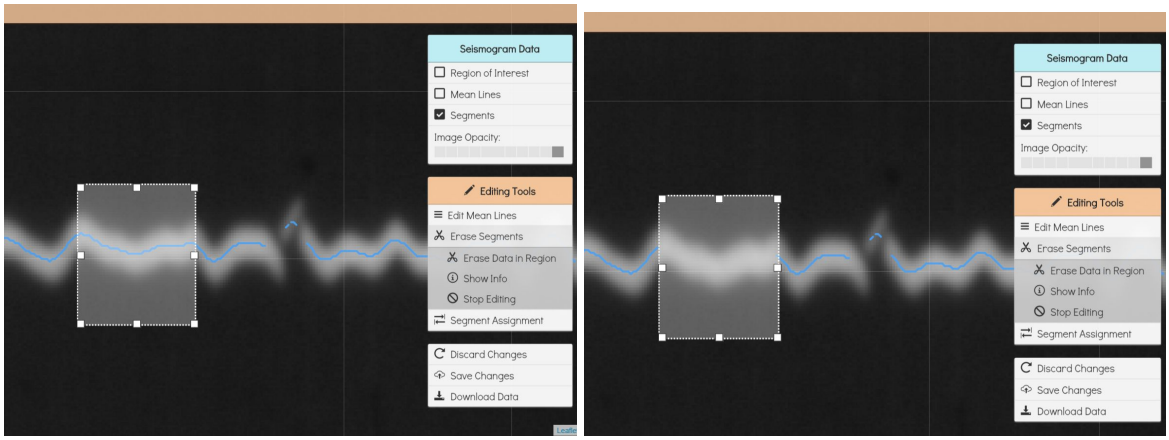
Images that are listed as "*Has Raw Data*" are those in which segments and centerlines have been automatically calculated. As it is with meanlines, the algorithm to calculate segments and centerlines is not 100% accurate, and typically errs by identifying noise and other superfluous features as segments. The editor allows the user to selectively delete these features[27].

To get into the segment editor, click on "*Erase Segments*." The following menu items appear. Clicking on "*Show Info*" will pull up the "*Segment Erasure Tool*" info box.



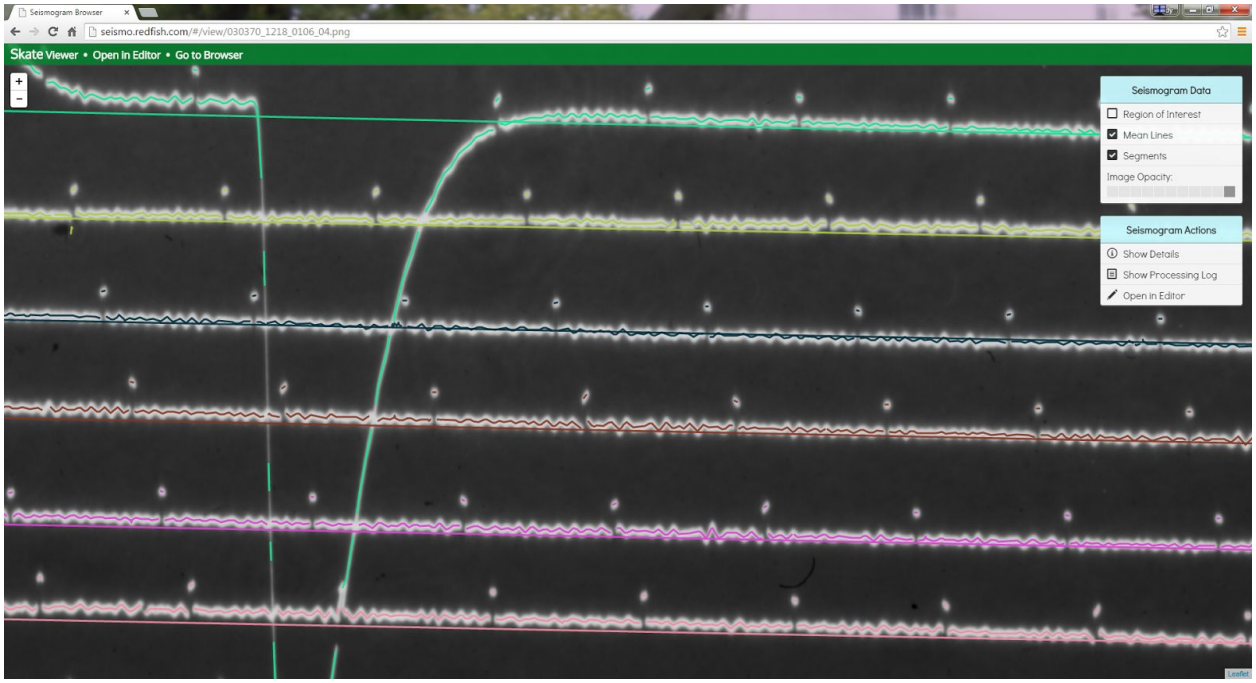
Once "*Erase Segments*" is clicked, an 8-handled rectangle selection tool will appear, as shown below. Zoom and pan as required to locate those segments to be deleted. Click and drag in the middle of the selection tool to move it around. Click and drag the selection tool's handles to adjust its size. When the segment to be deleted is enclosed by the selection rectangle, click "*Erase Data in Region*." The images below show before and after segment deletion.

[27] At this time there is no way to *add* segments and centerlines. Future improvements will likely allow this feature to be performed in data space rather than image space.

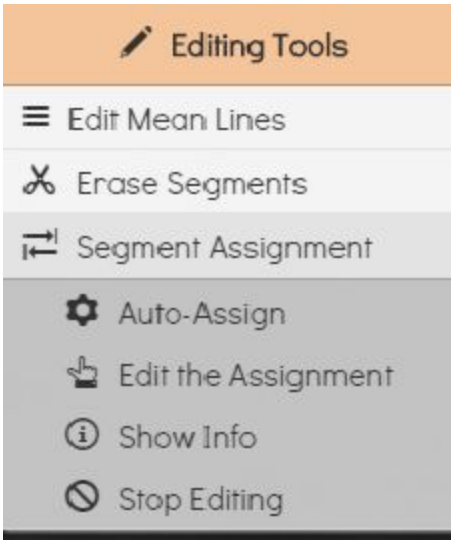


Segment Assignment

Once editing is complete, the Segment Assignment algorithm can be run by clicking on "*Segment Assignment.*" The algorithm uses the meanlines as a fundamental organizing feature. Therefore, after running the assignment algorithm, the segments will be colored according to the meanline with which they are associated, as shown in the image below.

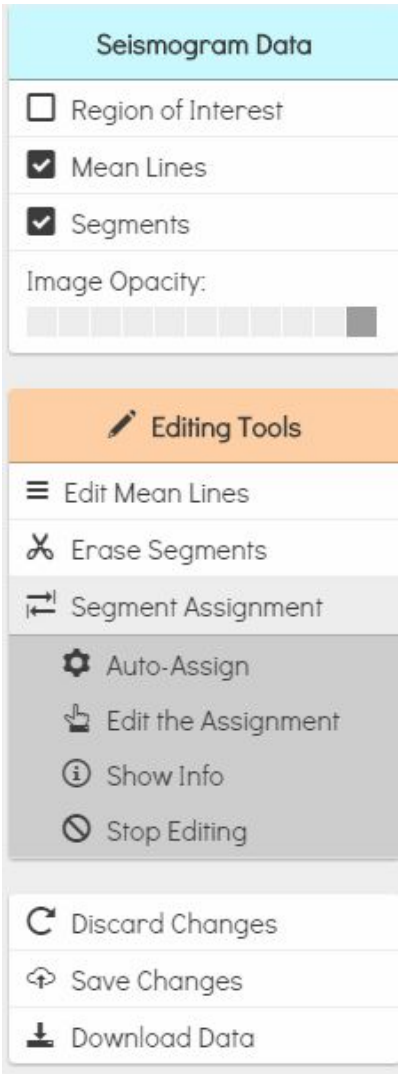


To run the assignment algorithm, click on "*Segment Assignment*" and then "*Auto-Assign*." While the time to run the algorithm varies, expect it to take 1-2 minutes. Do not click any other buttons while waiting for the auto assignment to complete.

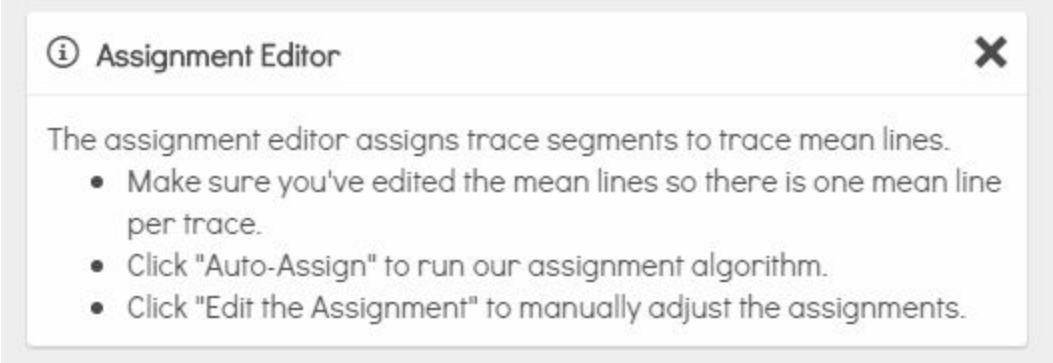


Editing Segment Assignment

Once the segment assignment algorithm has been run, the image can be inspected for accuracy, and edited again. Click on "*Segment Assignment*" and the following menu will appear.

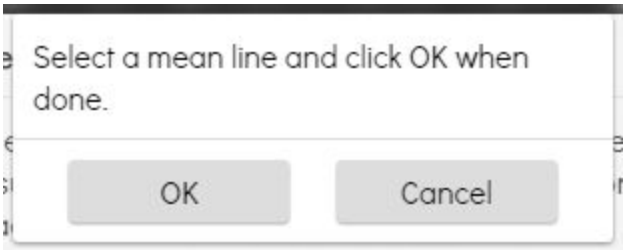


And clicking on "*Show Info*" will pull up the explanatory menu:



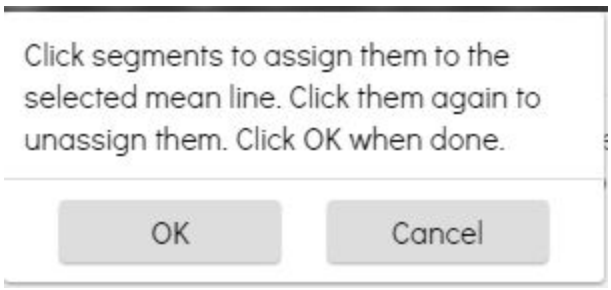
As the menu explains, it is important that one meanline is present and drawn to reasonable accuracy for each hour or partial hour of the seismogram[28]. This step should have already been performed during previous editing.

To change the assignment of a segment to a meanline, click on "*Edit the Assignment*." The following menu appears.



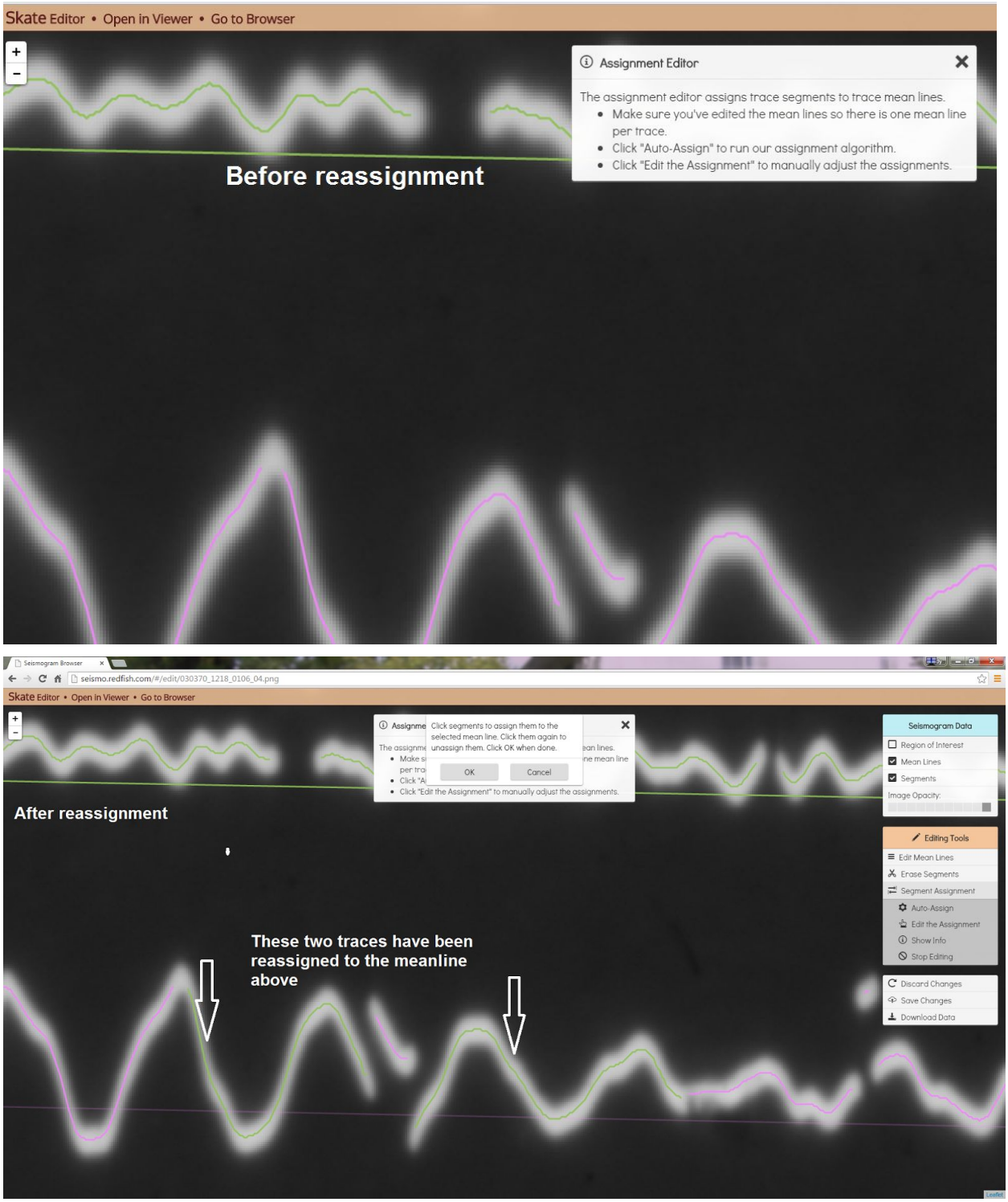
Click on the meanline to which you which to assign a misassigned segment, then click OK.

A new menu will appear.



Next, click on the segment to be reassigned, then click "*OK"*. More than one segment can be selected; continue to click on segments that will be reassigned. They will change color to match the meanline to which they will be reassigned as demonstrated in the image below (noting that the image shows an improper assignment, but nevertheless is shown here for instructional purposes). When selection is completed, click OK.

[28] Noting that we are currently only processing later date WWSSN long period seismograms, in which the length of a single line corresponds to one hour of recording time. Refer to the WWSSN Users Guide previously referenced for more information on seismogram characteristics.



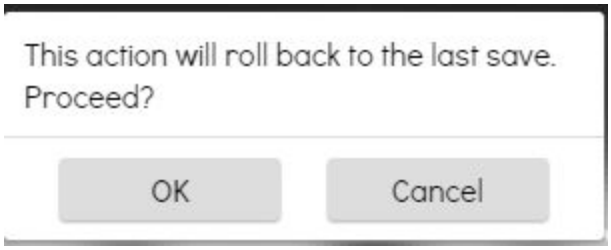
When reassigning is complete, the data can be saved or discarded. Also, the segment assignment algorithm can be run again in case the reassignment of segments changes the assignment of other segments. At this time we do not recommend running the assignment algorithm after editing the assignment.

Saving or discarding changes. Downloading data.

Important note on saving changes. SKATE is currently setup to store the raw data on S3; it will not be changed by any editing. However, changes to edited data will be permanently changed on S3; therefore the user needs to be very careful to make only those changes that are correct and necessary. As currently configured, the philosophy behind editing and saving is that multiple users can correct and improve any given seismogram; it is the availability of input from the broader seismic community that allows for the best result to be ultimately obtained. In addition, each save requires a transfer of data to S3, which incurs data transfer costs. Because there is a very limited operational budget, please limit your saving activities.

Discard Changes

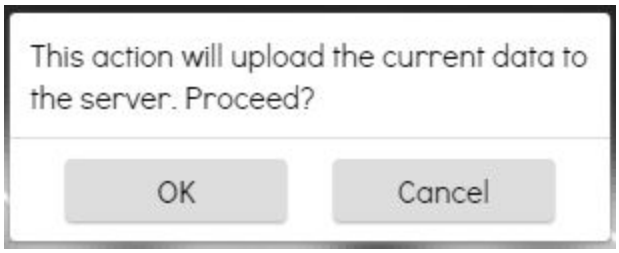
At any time any changes made can be discarded. This will revert the data back to the last saved value. Select "*Discard Changes*" and the following menu will appear.



Click "*OK*" to discard changes.

Save Changes

To save the changes, click on "*Save Changes*" and the following menu will appear.



Select "*OK*". This uploads the data to the S3 server.

Download Data

Once the seismogram is complete, the results can be downloaded to the user's machine by clicking on "*Download Data*". The data will be saved as a .zip file with the name of the seismogram record. You must be in Editor mode in order to access the download menu.

All of the data will be saved in a JSON format. Assignments will also be saved as a .csv file. Refer to www.json.org for details on the JSON format.

There are four JSON data files that are downloadable after analysis: roi.json; meanlines.json; segments.json; assignment.json. Their descriptions follow.

roi.json

The .json file looks like the following.

{"type":"FeatureCollection","features":[{"geometry":{"type":"Polygon","coordinates":[[[184,813],[14442,938],[14381,5555],[144,5431],[184,813]]]},"type":"Feature","id":null,"properties":{}}]}

The coordinates start in the upper left hand corner and follow a clockwise path around the image. Note that there are five coordinate pairs; the fifth is identical to the first, closing the polygon that defines the ROI.

meanlines.json

Meanlines are reported with an ID, and start and finish coordinates in an (x,y) format. The first few lines of the .json collection is shown below.

{"type":"FeatureCollection","features":[{"geometry":{"type":"LineString","coordinates":[[0,4356],[15456,4716]]},"type":"Feature","id":0,"properties":{}},{"geometry":{"type":"LineString","coordinates":[[0,4041],[15454,4378]]},"type":"Feature","id":1…}

Note that there may be gaps in the meanline ID numbering, and that they are not necessarily in a top to bottom order. This is a result of editing. Moving, adding or deleting lines will affect only the ID of the meanline being operated on, but will not affect other IDs.

The ID is an important identifier, as later it is used as the binding element to which all segments' centerline values are tied.

segments.json

Segments.json presents as:

{"type":"FeatureCollection","features":[{"geometry":{"type":"LineString","coordinates":[[10529,4713],[10530,4713]]},"type":"Feature","id":6769},{“geometry”…}

The “coordinates” are the (x,y) pairs of the centerlines for each segment. The coordinate system starts at the upper left of the image, with +y in the downward direction, and +x to the right. The “id” is the segment ID. This segment ID is used in conjunction with the meanline ID in the subsequently described assignment.json file.

assignment.json

The assignment.json file describes those segments that are associated with each meanline. The file looks like:

{"0":[segment ID 0-1, segment ID 0- 2,…segment ID 0-n],"1":[segment ID 1-1, segment ID 1-2,… segment ID 1-n], “N”:[segment ID N-1…segment ID N-n]}

The values in quotes are the meanline IDs. The values in brackets are the IDs of the segments belonging to that ID. The segment IDs are not necessarily sorted from left to right on the image.

assignment.csv

To download a .csv file of the assigned (i.e. connected) segments, choose ‘Open in Editor’ and select ‘Download CSV.’ The data is organized with column headers as: x0, y0; x1,y1;...;xN,yN. Each (xN,yN) pair corresponds to a single meanline, which in the case of long period WWSSN seismograms represents a one hour line of data. Note that typical spreadsheet plots, such as in Microsoft Excel, display positive Y values in Quadrant I of cartesian coordinate systems. However, the data in SKATE was obtained using Python, which plots positive Y values in Quadrant IV. Hence, Excel plotted data will be flipped about the horizontal axis with respect to the original image. Again, note that the meanline IDs will not necessarily be sorted from top to bottom, as editing and other actions might alter this sequence. Nevertheless, the data is easily parsed and represents the true output of the program.

mSEED

Not currently available, the data will be made available in mSEED format in the future.

Logout

At the end of the session, navigate to the home page by clicking on the “SKATE Viewer” link at the top of the screen. This link is only available when viewing a particular seismogram. Please logout when finished with the session.

Final

As discussed, SKATE is still undergoing significant development, and is not yet a complete digitization package. Future work will address the most important needs including, but not limited to: better meanline identification, improved rejection of spurious features, and the incorporation of advanced segment connection algorithms. Bug reports and comments on SKATE should be directed to Dr. Andrew Bartlett at andy@retrievertech.com.