SKATE: A web-based seismogram digitization tool

Article *in* Seismological Research Letters · August 2018

DOI: 10.1785/0220180006

CITATIONS
2

READS
178

5 authors, including:

Andrew Bartlett
Retriever Technology
31 PUBLICATIONS **432** CITATIONS

SEE PROFILE

ELECTRONIC SEISMOLOGIST

SKATE: A Web-Based Seismogram Digitization Tool

by Andrew H. Bartlett, Benjamin A. Lichtner, Marius Nita, Benamy Yashar, and Lowell E. Bartlett

ABSTRACT

Vast stores of paper- and film-based seismograms require novel solutions to extract their unique seismological data. We created a prototype software tool called SKATE (Software Kit for Automatic Trace Extraction) that is designed for high-speed, low-cost digitization of these archives. SKATE is a web-based program that requires a user only to navigate to a website (see Data and Resources) to begin digitizing seismogram images. The software is free, is server based, requires no software downloads, is open source, and has digitized up to the point of editing more than 30,000 World-Wide Standard Seismographic Network (WWSSN) images. It is based on a set of image-processing algorithms that identify real seismogram features while rejecting spurious noise and which attempts to assign these features to their proper place in a time series. Users can edit the results and download the data for further processing. The code is publicly available and is readily modifiable by qualified researchers.

INTRODUCTION

Vast archives of paper- and film-based historical seismograms contain unique seismological data that are currently unavailable to modern digital processing techniques. Extracting the data into a digital format from these archives, a process we refer to as digitization, will have a significant impact in many seismological studies, including those that address earthquake hazards, earth structure, and nuclear explosion monitoring.

Various computer programs have been developed for digitizing seismograms, and they have been applied to digitization of small numbers of seismograms (Bromirski and Chuang, 2003; Pintore, *et al.*, 2005; Bogiatzis and Ishii, 2016). Although successful in their approach, their labor and computational expenses prevent them from being effectively used for the digitization of large archives of data, particularly those with active and intersecting traces. Also, we have previously investigated image vectorization as a means to rapidly identify features in images of seismograms (Church *et al.*, 2013) based on proprietary software developed at Los Alamos National Laboratory (Prasad and Skourikhine, 2006). This approach, however, did not speed up the processing and required the use of closed source software. With these limitations in mind, our work has focused on the development of a software package that can rapidly digitize large archives at low cost. Additionally, we strove to create a program that was easy to operate and scalable and is open source.

With these goals in mind, we created the prototype seismogram digitization program called SKATE (Seismogram Kit for Automatic Trace Extraction). SKATE is a web-based image-processing program that is hosted on a remote Amazon Web Services (AWS) platform. Its use requires users only to navigate a browser to our user interface (see Data and Resources) to begin running the program. No software installation nor purchasing of software is required, and client computer requirements are minimal because all computationally intensive processing is done on the remote server.

Our goal of low-cost mass digitization is met with the use of powerful and scalable EC2 (Elastic Compute Cloud) instances and S3 (Simple Storage Service) webservices on AWS. To demonstrate this, we uploaded our entire repository of scanned World-Wide Standard Seismographic Network (WWSSN) seismographs—approximately 154,000 high-resolution.png images—and digitized up to the point of editing and placement of the traces in the final time series, more than 30,000 of these using Amazon EC2 cloud computing resources. This database of seismograms is fully searchable by date, type, and geographical location. The results are editable, and the outputted time-series data are downloadable both in JSON and CSV formats. We have made all seismograms in our database available for download at no charge and for any user. Initial digitization costs were less than $0.02 per image, a price that can readily be reduced in future iterations.

The user manual 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 already digitized seismograms.

Although the outputted data are not yet ready to be used in waveform analysis programs, SKATE has been developed to the point of a first release on our website. Many of the algorithms can be further tuned, and additional algorithms have been tested but not deployed, including the explicit identification of timing marks, spurious feature rejection, and trace connection routines. All of the code is open source and available at the GitHub website in Data and Resources, and interested researchers are encouraged to become contributors to this ongoing project.

 doi: 10.1785/0220180006

THE IMAGE-PROCESSING PIPELINE

The key tasks that SKATE performs are feature recognition, that is, the identification and separation from background of the seismic signal, the proper assignment of the signal to its place in the time series, and the editing of these data in image space. This is accomplished in an image-processing pipeline written in the open-source Python programming language. We created a comprehensive set of modules written in Python using a large library of existing image-processing functions that process the image and extract and output centerline, that is, time-series, data.

The pipeline is currently optimized to process WWSSN long-period images. The decision to focus on this particular set of data has allowed us to develop generic processes that can be modified for other types of images, including short-period WWSSN images, as well as others.

In this article, we use and briefly describe many common image-processing algorithms. There are many excellent monographs, including Gonzalez and Woods (2008), that readers can refer to for further detail. Additionally, web searches readily turn up many detailed definitions, including many excellent resources on Wikipedia.

The following sections detail the main algorithms in the pipeline in the order in which they are run.

Get Region of Interest

Most WWSSN seismograms written before mid-1978 are available as 70-mm film chips. As part of this project, a subset of these seismograms was electronically scanned and stored as lossless .png files. Consequently, the active or actual region of the seismogram, the region of interest (ROI), needs to be separated from the rest of the image, which includes its ID and unused white space. We also developed automatic ID identification using hit-or-miss techniques to identify the actual ID numbers on the image; this feature was not used in the work described in this article because file names were already associated with the images.

The algorithm uses a combination of intensity thresholding, morphological processing, and linear Hough transforms, combined with an expected value of ROI area for error checking, to yield a very accurate ROI. Images with ROIs outside the expected area value are flagged and processing is halted because this usually indicates a poor-quality image. Any features outside the ROI are not considered in later analyses.

Get Meanlines

What we refer to in this article as "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. Associating trace segments with their proper meanline, what we call trace assignment, ensures their correct placement in the final time series.

The meanline detection algorithm uses all the traces, active or not, as features in a linear Hough transform to determine meanlines. This works because all traces revert to the mean or very near and because distortions in the image are very small compared with the gap between hourly lines (Church *et al.*, 2013). Using timing marks for complementary meanline detection was tested but found to be much less accurate.

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.

Ideally, all long-period WWSSN seismograms used in this study should have 24 lines but midhour start and finish times, and seismograms that do not encompass the entire 24 hr 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 meanline positions because they are used only as a baseline from which trace assignment algorithms can proceed. As long as the meanline is closer to the correct hourly data than adjacent data, the distanced-based connection algorithms will operate as designed. A typical seismogram image with meanlines overlaid (color coded to show associations with associated segments, used in the editing process) is shown in Figure 1. Note that unevenly spaced meanlines are common. The shaded ROI is also visible in this image.

Flatten Background

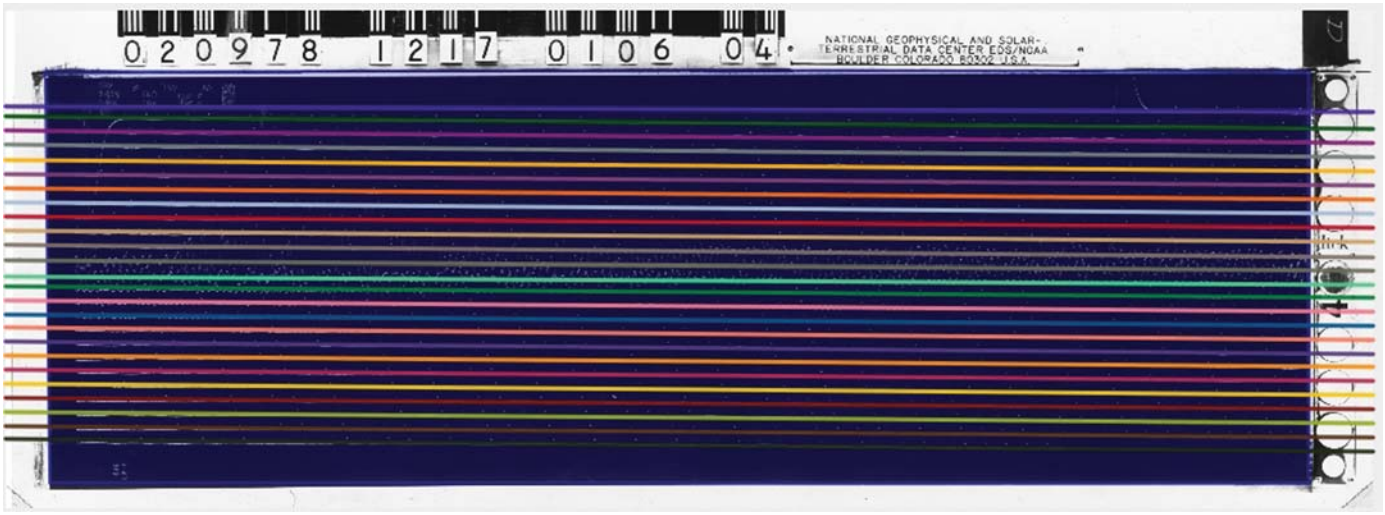
Because key parts of trace identification rely on local intensity changes, small changes in intensity at low intensity levels may lead to the identification of spurious features that might be confused with real signal. Background flattening and smoothing are therefore applied to assist in subsequent feature (trace) identification routines by removing low-intensity background fluctuations at both low and high frequencies. Flattening also is an important preliminary step in the segmentation of the image by creating "seeds" for subsequent watershed segmentation algorithms. A background that smoothly varies across the image is effective in dealing with images that have low-frequency intensity variations caused by, for example, film processing, storage, or other nonideal conditions.

Using histogram information, threshold parameters, and a bivariate spline fit, we identify and smooth the background. The result is a flattened image that has a smoothly varying background across the image.

Centerline Detection

This module is used to identify the centerline of the seismic traces, using a combination of spatial filtering tools. The goal is to find the single-pixel-wide path that defines the unique signal amplitude for each pixelated time increment: what we call the centerline. The centerline is different than what we call the "trace," which is the 2D image created by the light beam exposing the paper (Peterson and Hutt, 2014).

We apply a variety of spatial convolution kernels to identify the portions of the image that either are or are not a likely part of the centerline. Then, applying a variety of bitwise AND/OR operators to all the various convolved images, we identify the centerlines as only the likely centerline features



▲ **Figure. 1.** Computed meanlines and shaded ROI associated with a seismogram (vertical, long-period component from San Juan, Puerto Rico, starting at 12:17 UTC on 9 February 1978).

that are found in all the images, and we reject all others. Using multiple and complementary methods to identify centerlines greatly increases accuracy while minimizing spurious feature detection.

The spatial convolution kernels used are Laplacian, Sobel with directional dependence, and difference of Gaussians (DoG) with directional dependence over multiple levels of blur.

The Laplacian filter is an edge finding convolution. By keeping only positive values of the convolved image, we are able to identify the edges of the trace and assign these edges a FALSE, that is, noncenterline, value.

A challenge is to find the centerline of all traces over a wide range of intensity values and image scales. Quiescent traces have a high intensity owing to the relatively long residence time of the incident light beam on the original photosensitive recording paper (Eastman Kodak Company, 1970). Active traces are necessarily thinner and dimmer and consequently are noisier and more difficult to segment. We use the DoG filter to help with this problem.

The DoG filter is essentially a band-pass filter that identifies edge features at different scales and spatial frequencies. We create a 3D stack of images (what we call an image pyramid) to encompass different scales, comparing images at increasing levels of Gaussian blurs. The blurs are applied separately in both the vertical and horizontal directions, where the vertical is the conventional y direction in the original scanned image, and the x direction the horizontal, resulting in two pyramids each with directional dependence. When processing the 30,000 images that are available on SKATE, we used a 6-level pyramid to speed up the process and reduce compute costs; higher value pyramids can give better results.

Finally, a 1D 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 trace's intensity profile. The image is saved as a binary with the steepest slopes only having a FALSE 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.

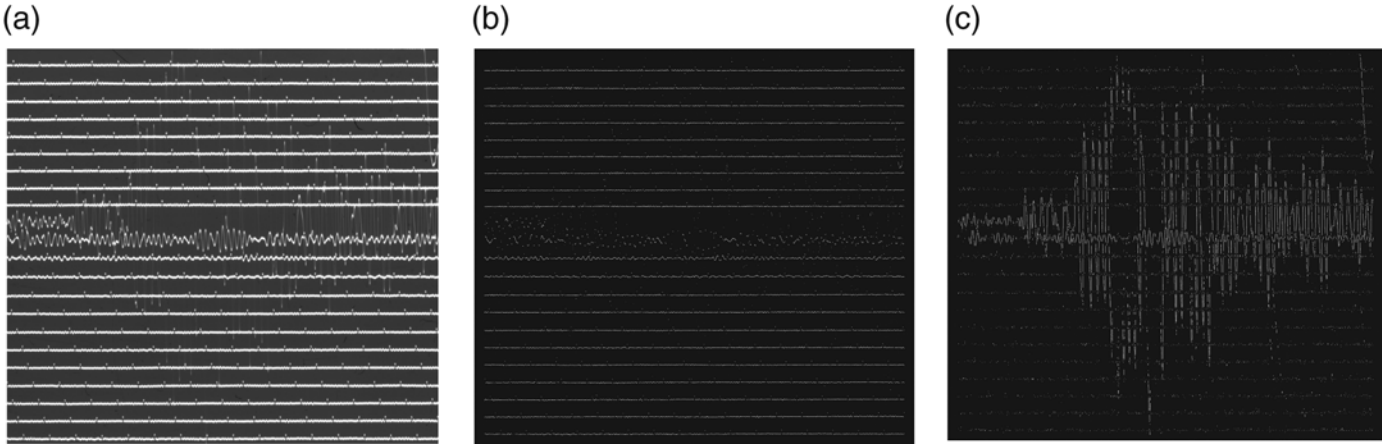
With the above series of images—Laplacian, Sobel, and DoG—a bitwise OR is now performed with the edge features determined from these kernels, as well as the seeds from the earlier flattening operation.

The returned image is a binary with those image features most likely to be associated with the centerline assigned as TRUE and the excluded background and edge regions assigned as FALSE.

Because the DoG was performed over six blur levels, we now perform the same bitwise OR operations on all scale levels and in both the horizontally and vertically dependent pyramids. The result is two 3D operators, each with respective horizontal and vertical dependencies—what we call the image cubes—that describe likely centerlines.

Because we are only looking for a single-pixel-wide centerline, we find local maxima at all levels in each image cube, 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. We then collapse each image cube into a single 2D array by selecting only the pixels from the local maxima that are TRUE across all levels of the image cube. Combining the results from the two cubes, the result is a 2D array of likely centerlines. We further limit the centerlines by imposing minimum length and connectivity constraints. These centerlines are saved and later will also be associated with actual trace segments in the image.

Typical results for horizontal and vertically dependent centerlines, which are then combined in the complete solution, are shown in Figure 2. The centerline images have been enhanced (eroded) to better display the results.



▲ **Figure 2.** (a) Original image (vertical, long-period component from College, Alaska, starting at 18:50 UTC on 25 March 1977). (b) Centerlines from the same region with horizontal dependencies. (c) Centerlines from the same region with vertical dependencies.

At this point, centerline data are not necessarily continuous, reflecting the conservative nature of this algorithm. Tuning the centerline detection parameters to add more centerline data while excluding noise can be readily accomplished by code improvements.

Segmentation

Although the above centerline detection determines the trace's path, we also need to have complete information on foreground information to find intersections and then to assign trace segments to centerlines.

Segmentation separates foreground data—the 2D seismic traces—from background. The algorithm accomplishes this by performing a watershed algorithm to segment the image into foreground and background features. The algorithm requires an input image and seed regions from which the watershed flooding will propagate until the flooding basins meet at the boundaries between traces and background.

Background seeds come from the previously described flattening procedure, plus those obtained from a morphological erosion of the input grayscale image using a structuring element sufficient to eliminate all traces. Foreground seeds use the previously defined centerline pixels, both vertical and horizontal.

The watershed algorithm is then run on the grayscale image, with small objects, both light and dark, eliminated with minimum size, length, and connectivity parameters. The output is a matrix with segmented regions identified with unique labels, representing trace features that include coordinates, intensity levels, and other useful feature information. These label values are used in the subsequent operations of intersection detection and centerline-to-meanline assignment.

Image Skeletonization and Intersection Identification

The previous segmentation procedure delivers the traces as objects. This segmented image is used to find the image skeleton, which in turn is then used to map the intersections where traces cross. The image skeleton is used instead of the centerlines because the former creates an image with the same connectivity as the original image versus the sparse connectivity of the centerline image.

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

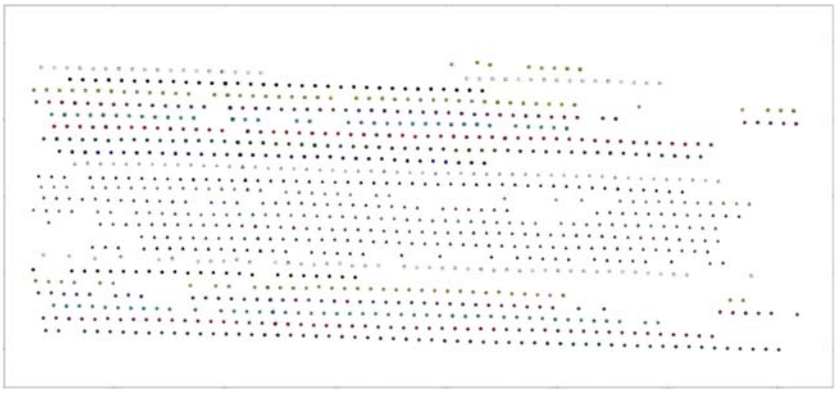
Identifying intersections is critical for creating accurate time series. When traces cross during seismic activity, we break the centerlines within the intersection because we cannot determine centerlines in the region when two traces overlap one another. Therefore, we locate intersections and eliminate the data within the intersection region in our final centerline data. These gaps can (but are not in this iteration of SKATE) later be bridged in the time-series data with curve fitting techniques.

Intersections need to be differentiated from dead ends, spurs, and false intersections. Spurs, which are short dendritic features forking off of the main skeleton, are removed by examining the ratio of the displacement of the spur's pixel path with respect to the width of the trace. Intersections are found by finding pixels in the skeleton 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 is found by getting the distance for the shortest path from each intersection to the edge of the trace using the distance transform of the image. This tends to overstate the size of the intersection (though not by much) but assures that trace centerline data are minimally confounded by adjacent, intersecting traces.

Each intersection is saved as a point with a radius value in a GeoJSON Feature Collection.

Trace Assignment and Timing Mark Identification

All previously identified segments and their associated centerlines are now assigned to a meanline. Currently, we are using a



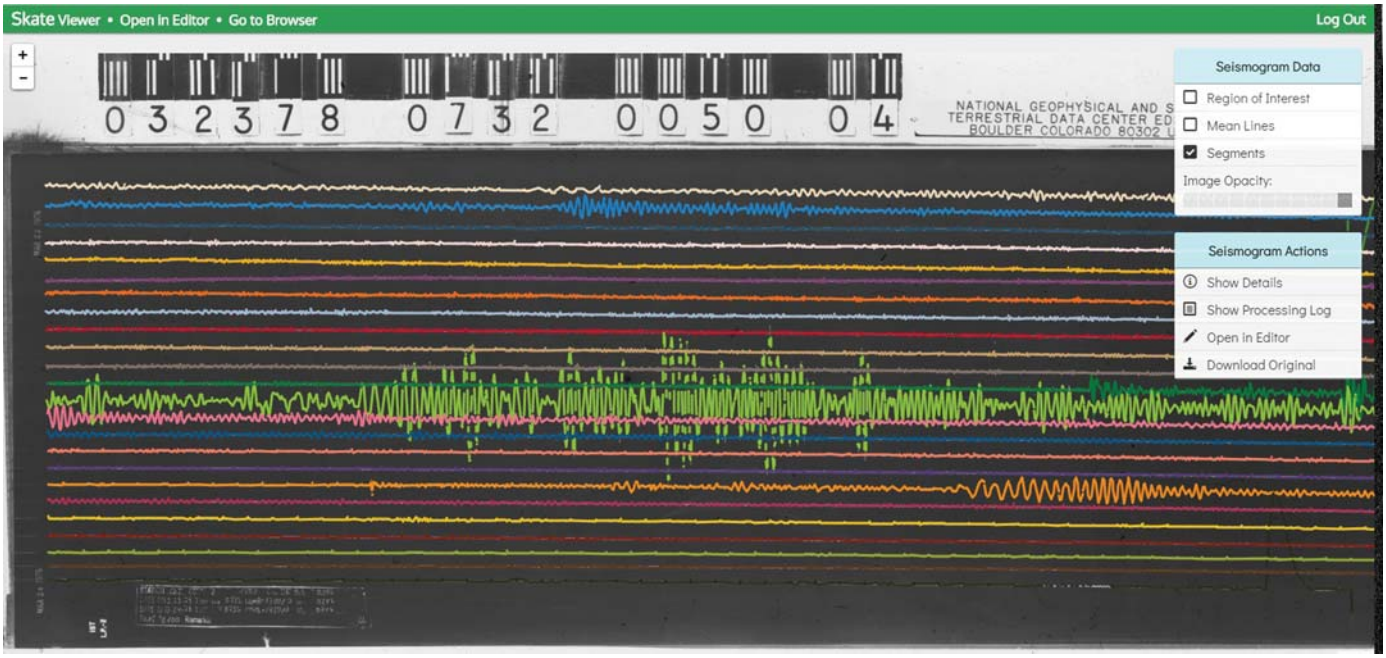
▲ **Figure 3.** Timing marks obtained from the assignment algorithm.

single algorithm that uses the previously collected segment data as inputs, including segment endpoints, centerline x values, y values, average y values, and standard deviation from segment mean. The algorithm uses the average y values of all segments to assign segments to the nearest meanline. 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 (~25% of the average vertical distance between meanlines): this is a parameter specifically tuned for WWSSN long-period images and chosen to capture most timing marks. During assignment, it builds a database of all of the meanlines, with information on their slope, which segments have been assigned to which meanlines, segment y distance from the meanline, and the domain in x that each assigned segment spans.

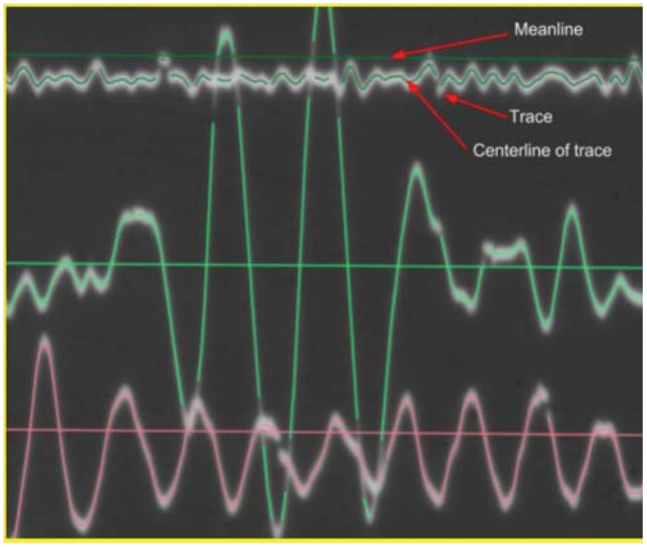
Next, timing marks are explicitly searched for by using a size and distance discriminator. Typical results are shown in Figure 3; the timing marks are then assigned to their appropriate meanline. Although not currently used explicitly, capturing a significant number of timing marks allows a map of timing mark locations to be created, which can later be used to normalize and connect timing marks to their associated trace.

Segments that were not initially assigned to any meanline are put into a list of orphaned segments. One by one, the assignment algorithm examines an orphaned segment's neighboring segments in the x direction that have already been assigned to meanlines. If the meanline of a neighboring segment has room for the orphan (i.e., if it has not already been assigned a segment that overlaps the orphan's domain), we assign the orphan segment to the meanline. We continue to iterate this process until all orphans are assigned or until their neighboring segments' meanlines have no more room. Remaining orphans are assigned to the closest meanline with room, regardless of neighboring segments. An additional benefit of this algorithm is that it partially eliminates spurious noise features by deleting features that cannot fit in any domain.

Figure 4 is an overall view of a digitized and edited seismogram. Figure 5 is a detail of the same seismogram with markers showing the traces, meanlines, and centerlines. Note that although the meanlines are not precisely placed, the assignment algorithm still worked well.



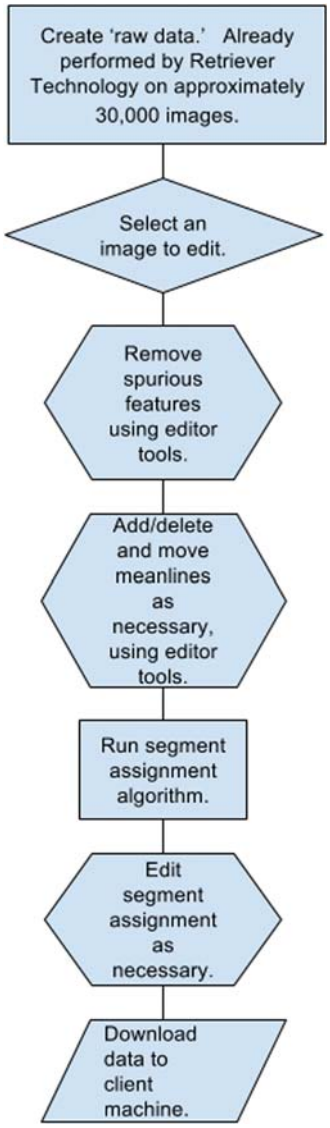
▲ **Figure 4.** Overall view of a digitized and edited seismogram (vertical long period from Istanbul, Turkey, starting at 07:32 UTC on 23 March 1978).



▲ **Figure 5.** Detail of Figure 4 with key features listed.

Editing and Data Output

When the assignment algorithm is complete, SKATE allows the user to edit meanlines, segments, and segment assignments in the browser. Detailed descriptions of the methodology are found in the user manual on the SKATE website; the steps in editing any seismogram are summarized in the flowchart in Figure 6. Several fully edited seismograms are available on the SKATE website that can be found by limiting the search to edited files. Users can also select any of the 30,000 processed seismograms, run the assignment algorithm, and edit them themselves. After editing, the centerline data are directly downloadable as CSV or JSON files and are listed as columnar x–y data for each hourly line on the seismogram.



▲ **Figure 6.** Flow chart of the editing process.

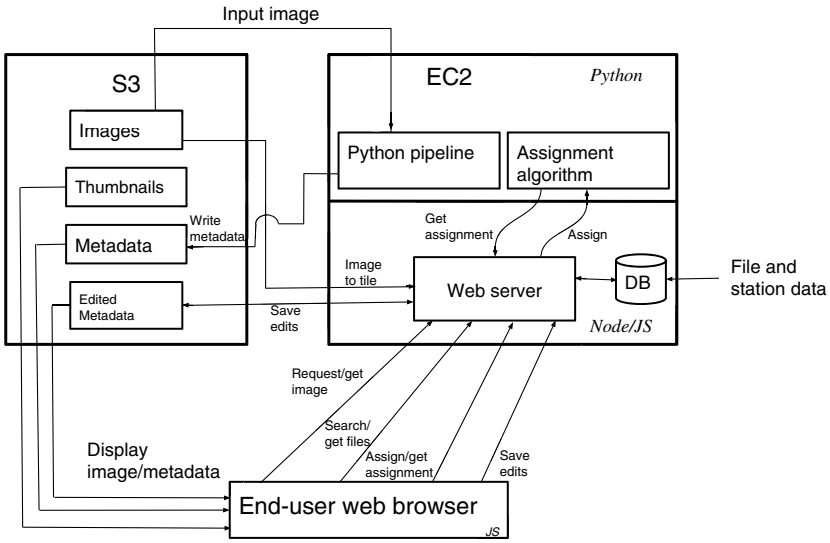
USER INTERFACE AND COMPUTER ARCHITECTURE

The user interface is a service-oriented web-based system that has been deployed on a host server using AWS. There are numerous reasons supporting this choice of architecture, including cost, reliability, ease of use, and scalability. The main features of the architecture are the image-processing pipeline, the browser app, and the server. The pipeline is the previously described set of Python image-processing algorithms. The browser app runs on the user's browser and consists of the file browser, the viewer, and the editing tools. It runs fully on the client side and interacts with the web server. When the user navigates to the user interface, the web browser downloads the entire web 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. The server is the code running on the remote host machine on AWS, up 100% of the time and ready to reply to requests sent to it by the browser app. The server can save metadata to S3, query the database for files, and verify authorization. It is the computational power behind the browser app.

The web server is implemented in Node.js. Node.js is a JavaScript runtime environment built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, nonblocking I/O model that makes it lightweight and efficient (see Data and Resources). It is scalable, allowing it to handle large volumes of requests that might occur in future development.

The browser app is written in JavaScript and 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 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



▲ **Figure 7.** Flow chart of the overall process.

scalability. It is well suited for our needs, especially with our simple data structures, which in this case consist only of file names and type, station IDs, date and time, and processing status. The 16-digit WWSSN filename was used to create most of these metadata.

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 the opening of large images. When viewing a seismogram, we only display the viewable part of the image, sampled down to match the current zoom level. This rendering technique dramatically speeds up browsing and editing activities, and it optimizes data transfer from AWS servers to the client computer. We use 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. A flowchart of the overall operation of the program is shown in Figure 7.

CONCLUDING REMARKS

Interpreting Department of Energy standards, SKATE is at a Technical Readiness level of 6 (demonstration in a relevant environment), with portions of the program, particularly the web interface, at near-final operational levels (U.S. Department of Energy, 2011). Practically, this means that the image-processing algorithms are in place and operational but need to be further improved in the following areas:

- Feature rejection. ROI border artifacts, handwritten notes and data stamps, and film processing byproducts can be dealt with using morphological, intensity, and assignment schemes.
- Feature recognition. As discussed, the methods used are conservative to limit postprocessing editing of spurious features. Improved recognition need not compete with feature rejection. Many parameters used in the pipeline algorithms can be better tuned or made user adjustable with small changes to the user interface.

In addition to the work presented in this article, we have investigated numerous other segment assignment algorithms, details of which are available at our GitHub repository. They include:

- Clustering methods based on Frey-Dueck and k-means techniques (Frey and Dueck, 2007).
- Multiple Traveling Salesmen using agent-based methods to find and reinforce the most likely paths across each hourly portion of the image.
- Bayesian methods with Kalman filters to identify the most probable connections for segments (Glickman and Van Dyk, 2007).
- Non-Subsampled Contourlet Decomposition: Directional decomposition of image with directional filter banks (Park *et al.*, 2000).

Detailed descriptions of these techniques, including alternate methods to map timing marks, can be found at the website in Data and Resources. Additional development efforts need to be taken to incorporate these into the software, but we anticipate that combining these methods in a multicriteria decision-making matrix to complete a trace path across each hourly portion of the seismogram will produce high confidence results.

The pipeline is written in Python and is freely available at our GitHub repository as a package of modules that can be run on a user's own computer. Previously discussed computing requirements may limit the size or resolution of the images being processed on a desktop computer, but these modules will allow developers to directly study and modify all of the parameters dictating the feature recognition quality in the outputted image. Editing is not available on the desktop version.

SKATE has been designed to gain acceptance and persistence in use by being written with all open-source software. We invite qualified researchers to help further develop and improve the code. Digitizing a seismogram in seconds on a website and pulling out a complete time-series solution in a very short time is unique and compelling and is the only practical way to unlock the information currently unavailable in millions of seismograms.

We are looking for partners to help us move the SKATE program and its 154,000 scanned WWSSN seismograms to a permanent site to ensure its long-term persistence. Continued development will allow it to meet nonproliferation national security goals and will expand the scope of seismology by opening up the vast archive of analog data to modern analytic techniques.

DATA AND RESOURCES

All data used in this article came from published sources listed in the references. All seismograms on the SKATE website and used in this article come from scanned World-Wide Standard

Seismographic Network (WWSSN) images provided to the U.S Geological Survey (USGS) under numerous contracts with Retriever Technology. The scans cover seismograms from the early 1960s up to the late 1970s and were chosen exclusively by the USGS. Digitized seismograms on the SKATE website were randomly chosen from 1970 and later. Long-period seismograms represent the vast majority of digitized images. SKATE is located at http://seismo.redfish.com. Ancillary information, including all software, additional documentation, and unpublished research, is found at https://github.com/Retrievertech. Users of the SKATE website who wish to save their own edited time-series data should contact Retriever Technology for login credentials. GitHub is available at https://github.com/retrievertech. Node.js is available at https://nodejs.org/en. Detailed descriptions of the techniques of GitHub repository can be found at https://github.com/retrievertech/Additional-documentation. All websites were last accessed on July 2018.

ACKNOWLEDGEMENTS

This work was performed under Department of Energy contract DE-SC000819, and we thank the Office of Science in providing the funding for this work. We would also like to thank Bob Hutt at the U.S. Geological Survey (USGS) for his assistance in this project and his dedication to preserving historic seismograms.

REFERENCES

Bogiatzis, P, and M. Ishii (2016). DigitSeis: A new digitization software for analog seismograms, *Seismol. Res. Lett.* **87,** no. 3, 726–736.

Bromirski, P. D., and S. Chuang (2003). SeisDig: Software to digitize analog seismogram images, user's manual, Scripps, *Institution of Oceanography Technical Report*, Scripps Institution of Oceanography, UC San Diego, 28 pp.

Church, E. D., A. H. Bartlett, and M. A. Jourabchi (2013). Raster-to-vector image analysis for fast digitization of historic seismograms, *Seismol. Res. Lett*. **84,** no. 3, 489–494.

U.S. Department of Energy (2011). *Technology Readiness Assessment Guide, DOE G 413.3-4A*.

Eastman Kodak Company (1970). *Kodak Linagraph 480 Paper*, Kodak pamphlet no. P-76, Kodak, Rochester, New York.

Frey, B. J., and D. Dueck (2007). Clustering by passing messages between data points, *Science* **315,** no. 5814, 972–976.

Glickman, M. E., and D. A. Van Dyk (2007). Basic bayesian methods, in *Topics in Biostatistics*, Humana Press, Totowa, New Jersey, 319–338.

Gonzalez, R. C., and R. E. Woods (2008). *Digital Image Processing*, Third Ed., Prentice-Hall, Inc, Upper Saddle River, New Jersey.

Park, S. I., M. J. T. Smith, and J. J. Lee (2000). Fingerprint enhancement based on the directional filter bank, *Proceedings 2000 International Conference on Image Processing*, Vol. 3**,** 793–796.

Peterson, J. R., and C. R. Hutt (2014). World-wide standardized seismograph network: A data users guide, *U.S. Geol. Surv. Open-File Rept. 2014-1218*.

Pintore, S., M. Quintiliani, and D. Franceschi (2005). Teseo: A vectoriser of historical seismograms, *Comput. Geosci.* **31,** 1228–1285.

Prasad, L., and A. Skourikhine (2006). Vectorized image segmentation via trixel agglomeration, *Pattern Recogn.* **39,** no. 4, 501–514.

Andrew H. Bartlett
Benjamin A. Lichtner
Marius Nita
Benamy Yashar
Lowell E. Bartlett
Retriever Technology
1600 Lena Street, STE D1
Santa Fe, New Mexico 87505 U.S.A.

Published Online 8 August 2018