Clustering and Visualization of Geodetic Array Data Streams using Self-Organizing Maps

Răzvan Popovici*
Răzvan Andonie†
Walter M. Szeliga‡
Timothy I. Melbourne‡
Craig W. Scrivner‡
*Altair Engineering Inc., Troy, MI, USA
and Department of Computer Science and Engineering, Oakland University, Rochester, MI, USA
† Computer Science Department, Central Washington University, Ellensburg, USA
and Electronics and Computers Department, Transylvania University of Braşov, Romania
‡ Department of Geological Sciences, Central Washington University, Ellensburg, USA

***Abstract*—The Pacific Northwest Geodesic Array at Central Washington University collects telemetered streaming data from 450 GPS stations. These real-time data are used to monitor and mitigate natural hazards arising from earthquakes, volcanic eruptions, landslides, and coastal sea-level hazards in the Pacific Northwest. Recent improvements in both accuracy of positioning measurements and latency of terrestrial data communication have led to the ability to collect data with higher sampling rates. For seismic monitoring applications, this means 1350 separate position streams from stations located across 1200 km along the West Coast of North America must be able to be both visually observed and automatically analyzed at a sampling rate of up to 1 Hz. Our goal is to efficiently extract and visualize useful information from these data streams. We propose a method to visualize the geodetic data by clustering the signal types with a Self-Organizing Map (SOM). The similarity measure in the SOM is determined by the similarity of signals received from GPS stations. Signals are transformed to symbol strings, and the distance measure in the SOM is defined by an edit distance. The symbol strings represent data streams and the SOM is dynamic. We overlap the resulted dynamic SOM on the Google Maps representation.**

I. Introduction

The Pacific Northwest Geodesic Array (PANGA) Geodesy Laboratory at Central Washington University (CWU) has a primary scientific role to support high precision geodetic measurements using Global Positioning System (GPS) observations in order to characterize crustal deformation, plate tectonic motions, coastal and earthquake hazards, and other environmental science applications. Under contracts from the National Science Foundation, the National Aeronautics and Space Administration, the U.S. Geological Survey, and UNAVCO, Inc., the Laboratory analyses all publicly shared GPS data within the Cascadia subduction zone [1] and greater Pacific Northwest. The Geodesy Laboratory analyzes data from roughly 1000 GPS stations that comprise the EarthScope Plate Boundary Observatory, whose stations span the Pacific-North American tectonic plate boundary from Alaska to Mexico. Fig. 1 is a map of all stations currently analyzed by CWU. In addition to serving as the Data Analysis Facility for the PANGA, the Geodesy Laboratory also supports field experiments on

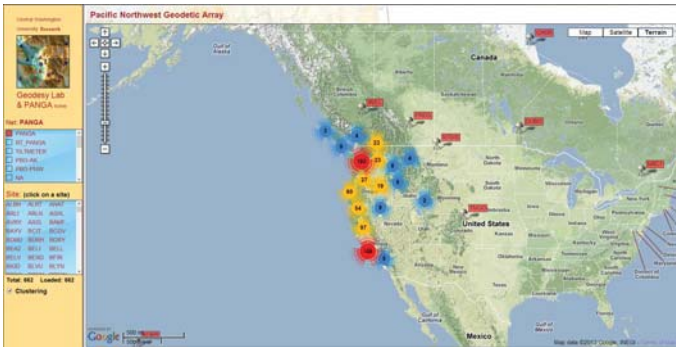

Fig. 1. Panga sensor placement.

Cascades volcanoes and in mainland Mexico, Baja California, California, Idaho, Montana, Oregon, and Washington. CWU also operates a continuous GPS network in Nepal.

Data from 450 GPS stations are telemetered in real-time back to CWU, where they are processed, also in real-time, using both NASA Jet Propulsion Lab's RTG [2] software as well as Trimble's RTKNet Integrity Manager software to provide relative positioning of several mm resolution across the Cascadia subduction zone and its metropolitan regions. These real-time data are used to monitor and mitigate natural hazards arising from earthquakes, volcanic eruptions, landslides, and coastal sea-level hazards. In addition, they are also used to monitor man-made structures such as Seattle's sagging Alaska Way Viaduct, WA SR-520 and I-90 floating bridges, and power-generation/drinking-water-supply dams throughout the Cascadia subduction zone, including those along the Columbia River.

The data streams from these 450 receivers is continuously downloaded, analyzed, archived and disseminated, as part of the existent geophysics and tectonics research programs within the Department of Geological Sciences, CWU. These tectonic displacement measurements are performed at millimeter-scale, and requires stringent analysis and parameter estimation techniques. The Geodesy Laboratory uses NASA's GIPSY

978-1-4799-4503-0/14/$31.00 ©2014 IEEE

OASIS (GPS Inferred Positioning SYstem, Orbital Analysis and SImulation Software) software to translate GPS satellite phase observables into position time series, and in-house parameter estimation and modeling software to quantify crustal deformation caused by plate tectonics, earthquakes, landslides and volcanic eruptions.

The position of a GPS station is estimated from a combination of satellite range and carrier phase measurements, with the position accuracy being heavily dependent on the accuracy of the satellite ephemerides, stability of the Cesium time standards on board the orbiting satellites, and the realism of models for both the tropospheric water content and electron density of the ionosphere.

Presently, such a continuously-operating, real-time network of GPS receivers is capable of detecting the strong ground motion that accompanies large earthquakes in real-time. For example, Fig. 2 shows data from one component of motion at a single site due to the well known M9 2011 Tōhoku earthquake in Japan. This time series is comparable to data from traditional seismic instrumentation for the event but is free from data artifacts common to seismic instruments, such as ”clipping” of a time series due to physical limitations of the instrument response. Thus, real-time high-sample-rate GPS solutions can provide useful data to determine the magnitude of large earthquakes in the immediate quake aftermath when the size of the event is still being determined and emergency response is being organized.

Ground deformation due to a major earthquake leads to a sudden change in the positions of sensors across a wide zone. One outstanding problem in monitoring any large network of continuously operating instruments is facile observation and analysis of its data streams. In the case of the Pacific Northwest Geodetic Array, the 450 GPS stations each output three continuous data channels: latitude, longitude, and vertical position. For seismic monitoring applications, this means 1350 separate position streams from stations located across 1200 km along the West Coast of North America must be able to be both visually observed and analyzed automatically. One of the characteristics of seismic events is spatial coherence of the observed earthquake deformation, which requires that station behavior be monitored in spatially-clustered groups.

There are two conflicting factors which motivate our present work. One is the potentially valuable information which can be extracted from the GPS sensors' data stream for detecting major earthquakes. Obviously, such data is valuable when extracted and processed in real-time. This takes us to the second factor, which is the challenge posed by mining big data sets of streaming data: we are more concerned about the abundance, not the lack of data. The recent advancements in data collection, such as data streams from sensors, exceed the ability of the data scientists to really put data in context of the questions and extract usable knowledge.

Our final goal is to make the real-time information from GPS sensors easily available. This includes wider public access via interfaces for all intelligent devices with a connection to the Internet. Practically, a geologist with mobile phone access, should have real-time access to streaming GPS position data. When combined with other measurements and information, this would help him to detect a major earthquake or landslide.

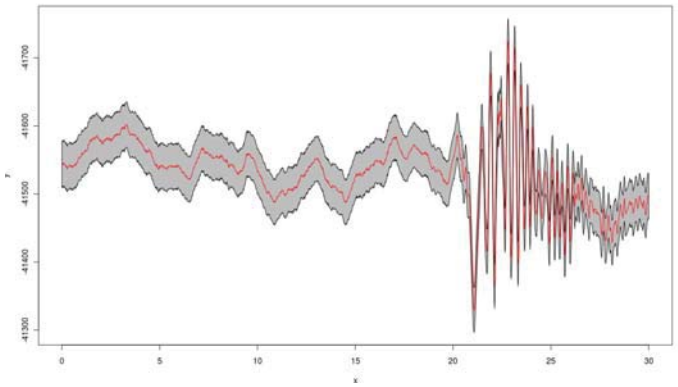

Fig. 2. Longitudinal movement of a GPS receiver located in Japan, during the Tōhoku earthquake. The read line is the reading and the gray area is one standard deviation border around the reading.

In this paper, we describe a novel clustering and visualization method for the evolution of geodetic data (latitude, longitude, and vertical position) received from the GPS stations. We are interested in clustering stations based on the similarity of their behavior, which is not necessarily determined by geographical distance. The input data streams are encoded into symbol strings. We use the edit distance to determine the similarity between strings. Based on this similarity, we cluster the data streams on a SOM. In the final stage, we overlap the resulted SOM on the geographical map (Google Map). The SOM is dynamic and updates periodically. This graphical representation of the data streams offers valuable information regarding the seismic changes. Our implementation is pluggable dashboard that monitors the real-time status of a network of GPS sensors.

The rest of the paper is structured as follows. Section II describes our input data and the preliminary processing steps. Section III explains the signal-to-symbol string transform. The SOM clustering model is detailed in Section IV, whereas Section V presents the graphical visualization of the results. Experiments, including on real earthquake data, are described in Section VI. Section VII contains the final remarks.

II. The input data

Our analysis runs on two data sets.

A. A PANGA data stream

The first data set is a six month capture of the life data stream provided by PANGA. We collected the data available real time from the PANGA network[1] for approximatively 6 months; on average we had $70 - 80$ simultaneously active sensors, with periods of one, two or five seconds. The sensors report geodetic data variations. This data set contains a few local earthquakes, but no major earthquake, visible on the GPS sensors charts. Each sensor may contain multiple streams, such as raw data and normalized data.

We queried the public PANGA API and parsed the JSON reply. Once obtained, the data has been stored in flat binary files, one file per sensor stream. The files have a simple

[1] http://www.panga.org/realtime/data/api

fixed size record format, composed of time stamp, variation on latitude, longitude and on elevation. We designed two methods of iteration for this data collection: *i)* iterate each stream in chronological order and *ii)* iterate the values of all sensors at once in chronological order. The first method is useful to compute signal statistics, with each sensor taken independently, while the second method serves as a simulation for real time dashboard, as it delivers all known data for a specific point in time.

B. Simulated earthquake data

The second data set simulates an earthquake on the same sensor network, using the Tōhoku patterns. This artificial data set is constructed by relocating, via a Helmert [3] transformation that includes a translation, reflection, rotation, and scaling of the Japanese national GPS network onto the Pacific Northwest geographical region, with the Tokyo metropolitan district aligned roughly to southwestern Washington State. This is motivated by the tectonic similarity of the two regions as seen from a seismological standpoint, both which are active subduction zones capable of releasing magnitude-9 sized earthquakes. After the Helmert transformation, the dynamic displacements from the M9 Tōhoku earthquake of 3/11/11 appear to propagate across the Cascadia region, simulating, to first order, displacements that might be observed during the next Cascadia magnitude-9 earthquake. This data set contains 846 sensors each with a maximum of 10,800 data points, recorded with 1Hz frequency. To reproduce the field data, some sensors are missing certain data points or intervals of time.

The data set does not retain the same format of PANGA – it is represented as space separated value text. This data has been saved in the same binary format as the above mentioned PANGA data, so that we benefited on the already implemented iterators we wrote for the initial data set.

III. Signals to strings

In order to transform a signal of length $L = nt$ in a string of length n, we break the signal in a set of time windows: $\{[0,t],[t,2t],[2t,3t],\ldots,[(n-1)t,nt]\}$ and we design a statistic which, applied to an interval, describes the trend of the signal. We identify the signal behavior within a given time windows by the following numbers, which are treated with respect of the string as letters: strong growth (2), growth (1), steady (0), decrease (-1), strong decrease (-2).

A. Quantifying the signal variation

Let $f : [t_1, t_2] \to \mathbb{R}$ be our signal, where $[t_1, t_2]$ is one of the $[(k-1)t, kt]$ intervals from above, and the meaning of the value is one of the three measurements (latitude, longitude, and elevation) for the respective point in time. We consider the signal continuous, but later we will move to the discrete case. Let $t_M = (t_1 + t_2)/2$, as shown in Fig. 3 be the middle of the interval.

We can consider the slope of the signal as our statistic. Unfortunately, the slope does not express the quantitative amount of variation, but rather a fold change. We want to compute a difference between the first and the second halves of the interval, as shown in Fig. 3. The design of our statistic is inspired from low pass filtering.

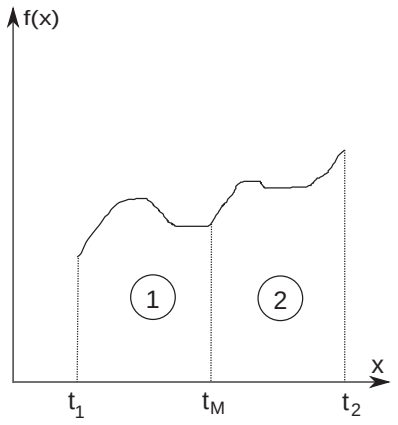

Fig. 3. Computing a statistic for signal clustering.

Let $w : [0,1] \to \mathbb{R}$ be a weight function, used to discriminate the values of f closer to the point t_M against the ones closer to t_1 and t_2. We extend this function as:

$$\begin{aligned} &w' : [-1,1] \to \mathbb{R} \\ &w'(x) = \begin{cases} w(x), & \text{if } x > 0. \\ -w(x), & \text{if } x < 0. \\ 0 & \text{if } x = 0. \end{cases} \end{aligned} \tag{1}$$

Obviously, $w'(x) = -w'(-x)$.

We compute the weighted difference between areas 1 and 2 (Fig. 3):

$$\begin{aligned} \Delta_{t_1}^{t_2} &= \int_{t_M}^{t_2} f(x) w\left(\frac{x - t_M}{t_2 - t_M}\right) dx - \int_{t_1}^{t_M} f(x) w\left(\frac{t_M - x}{t_M - t_1}\right) dx \\ &= \int_{t_M}^{t_2} f(x) w'\left(\frac{x - t_M}{t_2 - t_M}\right) dx + \int_{t_1}^{t_M} f(x) w'\left(\frac{x - t_M}{t_M - t_1}\right) dx \\ &= \int_{t_1}^{t_2} f(x) w'\left(\frac{x - t_M}{t_2 - t_M}\right) dx \end{aligned} \tag{2}$$

where: $t_2 - t_M = t_M - t_1 = (t_1 - t_2)/2$.

With $w(x) = x$, the statistic becomes:

$$\Delta_{t_1}^{t_2} = \int_{t_1}^{t_2} f(x) \frac{x - t_M}{t_2 - t_M} dx \tag{3}$$

For the discrete case, assuming that we have $x_1, x_2, \ldots, x_n$ discrete values measured at times $t_1, t_2, \ldots, t_n$, eq. (3) can be written as follows:

$$\Delta_{t_1}^{t_n} = \frac{1}{n-1} \sum_{i=1}^{n} x_i \frac{t_i - t_M}{t_n - t_M}, \tag{4}$$

were $t_M = (t_1 + t_n)/2$.

In the worst case, when only the values from extremities of the interval are available, we have $t_M = (t_1 + t_2)/2$ and

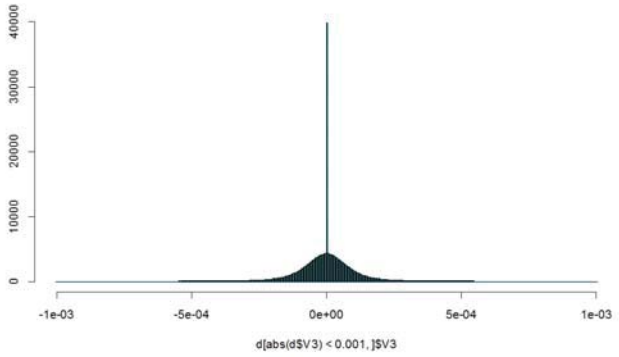

Fig. 4. Distribution of the $\Delta_{t_1}^{t_2}$ statistic on non-earthquake data.

$$\Delta_{t_1}^{t_2} = x_1 \frac{t_1 - t_M}{t_2 - t_M} + x_1 \frac{t_2 - t_M}{t_2 - t_M} = x_2 - x_1 \tag{5}$$

The statistic can be computed on two vectors of different size, but corresponding to an equal time window. Other works, such as [4], allow comparison of different length strings and handle the difference through the signal comparison distance.

B. Distribution of the $\Delta_{t_1}^{t_2}$ statistic

After computing the $\Delta_{t_1}^{t_2}$ statistic by eq. (4) from non-earthquake signals of 30 seconds length collected during six months from the PANGA network, described in Subsection II-A we have obtained the distribution shown in Fig. 4. We have to note that $T_n - T_1 = 30s$ but n may vary due to different sampling rate. By removing the predominant value of 0, we observe that our distribution is normal, with standard deviation $\sigma = 0.01$. Running the same statistic on simulated earthquake data, we find the earthquake data, mentioned in Subsection II-B having a much larger variance, with $\sigma = 10$. In both cases, the variance is the same on the three measurements directions: north, west, and elevation. Also, the mean of the statistic is zero.

C. Number assignment

We assign a number q between -2 and 2 to the signal, based on the computed statistic $\Delta_{t_1}^{t_2}$ and the observed standard deviation σ, as follows:

$$q = \begin{cases} -2, & \text{if } \Delta_{t_1}^{t_2} < -2\sigma. \\ -1, & \text{if } \Delta_{t_1}^{t_2} \in [-2\sigma, -\sigma) \\ 0, & \text{if } \Delta_{t_1}^{t_2} \in [-\sigma, \sigma] \\ 1, & \text{if } \Delta_{t_1}^{t_2} \in (\sigma, 2\sigma] \\ 2, & \text{if } \Delta_{t_1}^{t_2} > 2\sigma \end{cases} \tag{6}$$

IV. The SOM clustering of signal strings

Once we transform the signals to strings of the same size, we aim to cluster our input data and determine the centroids associated with various patterns of signals. Instead of the standard SOM for strings [5], we will use the ESOM [6], which is specifically designed for on-line data streams because *i)* it is quickly able to assign a new cluster to a newly occurred pattern, and *ii)* it can discard clusters if no signal fulfills the minimal distance to the centroid within a prescribed number of inputs.

Our contribution is to embed our application into the ESOM framework, which turns out to be quite laborious. First, we have to define the distance between two strings. Second, we have to define the learning process in the ESOM.

A. The distance between strings

Our starting point is the Levenshtein (edit) distance between two strings [7]. The edit distance is sensitive to displacements in the strings, matching the common pattern even it is not located at the same offset relative to the base, being superior, in the context of pattern matching, to covariance based distances. The Levenshtein distance is computed as a sum of costs, associated with the need to change, delete and insert a character. The original Levenshtein distance is not sensitive to how different two characters are. For example, turning a letter into another letter has a fixed cost, no matter the letters.

In our case, the numbers correspond to different classes of variation. For example, replacing -2 with -1 shall carry a smaller cost than replacing -2 with 1. Another observation is that class 0, associated to a steady signal, is neutral with respect to the information from that signal. Hence, the cost of inserting or deleting a 0 should be less than the cost of the same operation performed with another character. For similar reasons, Needleman and Wunsch [8] introduced a dissimilarity matrix, to describe the replacement cost (which can be different) for each two characters.

To formalize this concept, we introduce the alphabet $Q = \{-2, -1, 0, 1, 2\}$ and the the Kleene closure of Q (i.e., the set of all strings over Q of any length). Let ϕ be the empty string. We can design our cost function as the absolute value of the difference of two numbers.

We consider the following $lev : Q^* \times Q^* \to \mathbb{R}^+$ modified Levenshtein distance:

$$\begin{aligned} lev(a,b) = min\big(&lev(a',b') + |a_n - b_m|, \\ &lev(a',b) + |a_n| + \delta, \\ &lev(a,b') + |b_m| + \delta\big) \end{aligned} \tag{7}$$

where: $a, b \in Q^*, a = (a_1, a_2, \ldots, a_n), b = (b_1, \ldots, b_m)$ and the condition to end the recursion is $lev(\phi, \phi) = 0$. The strings a' and b' are obtained by removing the last element from strings a, respectively b. In the above recursive definition, a' and b' are not defined for $a = \phi$, respectively $b = \phi$. δ is a positive constant, representing the costs of one insertion or deletion. The function lev is a distance defined on the space Q^* of strings.

Essentially, our modification consists in introducing the variable cost terms $|a_n| + \delta$ and $|b_m| + \delta$ for insertion, respectively and deletion. The Needleman-Wunsch distance [8] prescribes a fixed cost in these cases.

The distance lev is able to distinguish between the intensities of the signal, it penalizes much more insertion and deletion

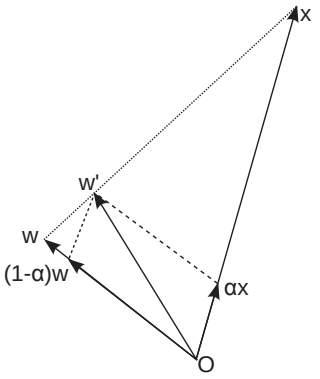

Fig. 5. Relation between x, w and w'.

of extreme values than insertion of steady (0) or close to steady signals. The same approach works for modification: neighbor signal classes yield less distance than the opposite ones.

We defined the distance between signals in this simplified manner because the meaning of the five signal classes we operate with is well known. However, if the nature of the signals and the relations between them would be unknown, a dynamic dissimilarity matrix may be employed, as in [9].

B. Training the network by adjusting strings

In [6], the nodes of the ESOM are vectors, with a well defined vector operations (addition, subtraction and multiplication with scalar) as well as the norm operator. The update algorithm is governed by eq. (8), where w is a centroid, x is an input pattern, and w' is the updated value of w. The parameter γ is a small learning rate and σ controls the effective neighborhood spread.

$$w' = (1-\alpha)w + \alpha x \text{ where } \alpha = \gamma e^{-\frac{\|w-x\|}{\sigma^2}} \tag{8}$$

To be able to employ the training algorithm to operate with strings, we have to adapt the formula so that it only uses the distance, which is the only operator available on the string space. First we replace the norm of the vector difference with the distance between two elements, in our case, it is the adapted Levenshtein distance described in Subsection IV-A. In a strictly convex Banach space, we can prove that eq. (8) is equivalent to eqs. (9) and (10), where $\alpha = \gamma e^{-\frac{lev(w,x)}{\sigma^2}}$.

$$lev(w,x) = lev(x,w') + lev(w',w) \tag{9}$$

$$\alpha = lev(w,w')/lev(x,w) \tag{10}$$

The string space is not a strictly convex Banach space, but the resulted pair of equations operates with distances only, therefore they provide an approximation good enough for the adjusted value of the element, we need to compute.

The reasoning behind the assumption is depicted in Fig. 5, which is a geometrical interpretation of eq. (8). In this representation, our strings have a vector representation, which is mathematically not correct, but intuitive. Using the parallelogram rule, we obtain w' as the sum of the vectors $w_1 = (1-\alpha)w$ and $x_1 = \alpha x$. Following, we demonstrate that w,w' and x are collinear. Since $\angle ww_1w' = \angle wox$, because $w_1w' \parallel Ox$, $ww_1/Ow = (Ow - Ow_1)/Ow = (Ow - (1-\alpha)Ow)/Ow = \alpha$ and $w'w_1/Ox = Ox_1/O_x = (\alpha Ox)/Ox = \alpha$. Therefore, $\triangle ww_1 \sim \triangle ww_1w'$, according to the side-angle-side similarity criterion, with the ratio α. Because of this similarity, $\angle Owx = \angle Oww'$. Therefore, w, w' and x are collinear, and $ww'/wx = \alpha$.

While we can easily find strings which fulfill eq. (9), due to the discrete nature of the string space, we cannot always find w', so that it fulfills eq. (10) for an arbitrary α. Therefore, we shall identify the w' strings which fullfil eq. (9) and, at the same time, minimize the approximation error in eq. (10).

Given two strings w and x, we use the Needleman-Wunsch algorithm [8], based on dynamic programming, to compute both the Levenshtein distance between the two strings and the minimum sequence of operations to transform one string into the other. These operations are from the set $R = ins(q)$ (insert), $del(q)$ (delete), $chg(q,p)$ (change q into p), and $keep(q)$ (do no change, just advance the next character), where $q, p \in Q$. Let us consider R^*, the Kleene closure of R.

Data: $r \in R^*$
Initialize x with the empty string;
for *each element* $op(q_k)$ *in r:* **do**
  if $op \in \{keep, chg, ins\}$ **then**
    append q_k to x
  end
end
Result: x

Algorithm 1: Algorithm to compute x when the r sequence of operations is known.

Algorithm 1 shows how the optimal sequence of operations leads to the goal string x. We change this algorithm, so that it builds a different string, as well as an annotation which will serve as support for the construction of the solution w'.

The changed Algorithm 2 produces a string y where the deletions are not being applied. For each character of the newly produced string, we compute the corresponding entry in the boolean vector $removable$, which holds $true$ for characters originated of either insertions or skipped deletions, and $false$ for the rest. Also, we build an additional vector of ranges. The ranges are computed as following: for insertions and deletions, they are between the subject character and the median character 0, changes ranges are between the original and the new character and the $keep$ operations yield no range.

We observe that not only the string resulted from Algorithm 2, but also any string produced by removing characters corresponding to $true$ values in $removable$, as well as alteration of the remaining characters, by picking a value from the range, satisfies eq. (9). We can prove that the string generation method is exhaustive: no string satisfying eq. (9) can elude our generation method.

We have to find now the strings which also minimize the approximation error for eq. (10). We solve this search problem by backtracking, which is acceptable for our small search space. Due to the fact that the distance between strings is a combination of unit and δ additions, a more computational

Data: $w \in Q^*$, $r \in R^*$
Initialize y with the empty string;
Initialize $removable$ with the empty vector;
Initialize $ranges$ with the empty vector;
for *each element* $op(q_k, p_k)$ *in r:* **do**
  append q_k to y;
  $start = q_k$;
  if $op \in \{ins, del\}$ **then**
    append $true$ to $removable$;
    $stop = 0$;
  else
    append $false$ to $removable$;
    $stop = p_k$;
  end
  $r = [min(start, stop), max(start, stop)]$;
  append r to $ranges$;
end
Result: y, removable, ranges

Algorithm 2: Algorithm to compute y when w, and the r sequence of operations are known.

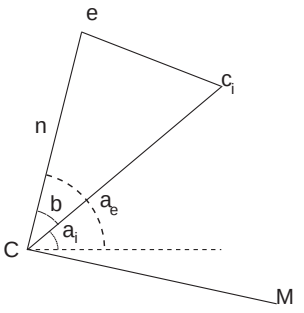

Fig. 6. Computation of the color angle.

efficient approach could be used, if needed. From the set of solutions, we choose the smallest string w' (in lexicographic order).

V. Graphical representation and implementation

Once the SOM is being constructed, we discard the neighborhood relations and assign colors to the centroids. Based on the distance from the closest centroid and the neutral point, each point of the string space can be colored. For the neutral string, composed integrally of 0s, we assign the white color.

We assign an angle a_i from the set $\{0°, 30°, 60°, \ldots, 330°\}$ to each cluster centroid, c_i. The allocation is performed by the dimension multidimensional scaling (MDS) [10], [11] algorithm.

For a given string e, we identify the closest cluster centroid, with respect of our modified Levenshtein distance, from eq. (7). We take one of the elements of the set: $\arg\min_{c_i} lev(c_i, e)$.

We determine the distance between e and the neutral element C = "0000000000". We compute the angle a_i determined by e, C and c_i, given that the edges of the triangle generated by the three points can be computed, as shown in Fig. 6.

Let M be the maximum of the distances from signals to the center C. The r_e is a normalized value, it belongs to the $[0, 1]$ interval. Given a_e and r_e, we can compute the color of the element by assigning the angle a_e to hue and the normalized $r_e \in [0, 1]$ to saturation.

$$b = \arccos\Big(\frac{lev(C,c_i)^2 + lev(C,e)^2 - lev(c_i,e)^2}{2lev(C,c_i)lev(C,e)}\Big)$$
$$a_e = a_i + b,\ r_e = \frac{lev(C,e)}{M} \quad (11)$$

We realize the graphical representation of the clustered GPS stations (Fig. 7), which is drawn over the map using: Google Maps and Overlay layer to render the map, JSON format for colors and sensor positions, and d3.js for SVG rendering within the overlay layer. The visualization is completely written in JavaScript, and it does not require any server component, as long as the JSON data file is being provided to the application.

TABLE I. SOM coefficients for different runs.

Run	ϵ	σ	λ	$MaxGen$	$MaxDist$	N
Toy 1	9	$\epsilon/2$	0.02	1000	25	5
Toy 1	9	$\epsilon/2$	0.02	1000	20	5
Toy 3	13	$\epsilon/2$	0.02	1000	20	5
Toy 4	6	$\epsilon/2$	0.02	1000	20	5
Toy 5	9.1	$\epsilon/2$	0.02	1000	50	5
Earthq.	10	$\epsilon/2$	0.2	100000	50	5

VI. Experiments

We perform a number of experiments, first on a toy example, then on the earthquake data.

The parameters of our ESOM implementation are:

- ϵ is a error threshold, used to define the concept of neighborhood; it is expressed in the distance unit and defines whether the newcomer elements goes in an existing cluster or it will have its own new cluster.
- σ controls the neighborhood spread.
- λ controls the learning rate of the neural network, the impact of new elements against the old centroids.
- $MaxGen$ is being introduced as a response to the requirements to prune inactive nodes, according to the 4th step of the ESOM algorithm from[6]. If the node has not been adjusted or linked the provided number of learn cycles, it is being removed after exceeding the value.
- $MaxDist$: same as above, this coefficient has been introduced as result of need to prune weak connections, from the same step in ESOM algorithm as $MaxGen$. If the distance of the two elements exceeds the coefficient (due to incremental application of the learning mechanism) the connection is being deleted.
- N controls the number of neighbors initially assigned to an elements. In the original description [6], N is hardcoded at 2, while we allow any value to be provided. If N is chosen excessively large, so that a new centroid starts with many initial connections, it is likely that these connections will be removed upon applying $MaxDist$.

We design a simple experiment with five well-know and easy distinguishable signal types: steady, increases than decreases, decreases than increases, constantly increases and steadily decreases. These signals don't originate from any data set described in Section II, but they were manually generated, a few examples have been provided for each class. We vary the values of the coefficients according to Table I and observe the topology of the resulting network. For brevity, the diagrams use the letters A to E instead of the numbers -2 to 2.

- For the first case shown in Fig. 8, all centroids were identified, however it occurs twice that an expected cluster is represented by two centroids and also, there are neighborhood relations between opposed patterns, such as steady growth and steady decrease.
- In the case of Fig. 9, the problem of multiple centroids persists, however only applied to one cluster, and some expected topological relations are missing, but there is no relation between opposing patterns.
- The third run (Fig. 10) fails to identify two of the centroids.
- Fig. 11 depicts the fourth case, where additional centroids for each cluster are being returned, while properly identifying the topology.
- The fifth run (Fig. 12) identifies properly the centroids of each cluster and shows 5 out of the expected 6 neighborhood relations between the clusters.

More relevant, but harder to follow is the analysis of the PANGA simulated earthquake data described in Subsection II-B, using the parameters from Table I. The resulted topology is shown in Fig. 13 and the dynamic graphical representation of the clustered GPS sensors can be accessed online[2].

VII. Conclusion

We have described a clustering and visualization method for signals from GPS stations. The output depends largely on the a priori knowledge of the standard deviation of Δ statistics associate to the signals as well as on the choice of coefficients of the neural network, which basically control: *a)* the number of generated clusters and *b)* the speed of forgetting past data irrelevant in the present.

The principal application of the algorithm presented here is in the real-time monitoring of large, dense, and spatially extensive geodetic networks. Current and incipient networks will run into the thousands of stations, preventing manual inspection of data analysis for transient phenomena such as that caused by earthquakes, landslides, or other natural hazards. Much as seismic networks are currently monitored for exceeding of floor noise levels to trigger the identification of an earthquake, the clustering algorithm outlined here can be utilized to provide automated transient detection for alarming and monitoring systems.

Our method can be plugged in a real time seismic data processing pipeline, or it can be used as a visualization tool for relevant seismic events. The method will incrementally learn new patterns and dynamically update the SOM and the graphical representation.

[2]http://www.miravtech.com/maps/map.html

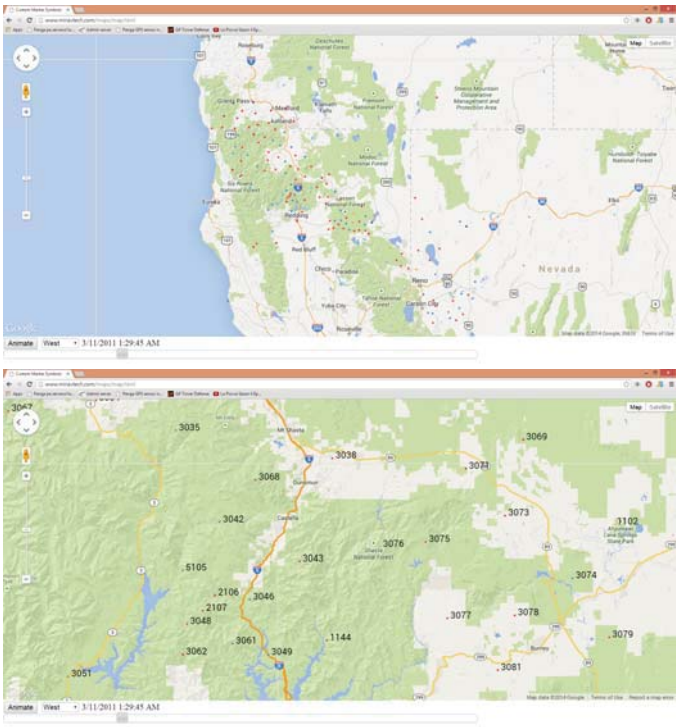

Fig. 7. Visual representation of the sensor network.

References

[1] A. C. Aguiar, T. I. Melbourne, and C. W. Scrivner, "Moment release rate of Cascadia tremor constrained by GPS," *Journal of Geophysical Research*, vol. 114, Jul. 2009.

[2] W. Bertiger, S. D. Desai, B. Haines, N. Harvey, A. W. Moore, S. Owen, and J. P. Weiss, "Single receiver phase ambiguity resolution with GPS data," *Journal of Geodesy*, vol. 84, pp. 327–337, May 2010.

[3] G. Watson, "Computing helmert transformations," *Journal of Computational and Applied Mathematics*, vol. 197, no. 2, pp. 387–394, 2006.

[4] A. Marzal and E. Vidal, "Computation of normalized edit distance and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 9, pp. 926–932, sep 1993.

[5] T. Kohonen and P. Somervuo, "Self-organizing maps of symbol strings," *Neurocomputing*, vol. 21, pp. 19 – 30, 1998.

[6] D. Deng and N. K. Kasabov, "ESOM: An algorithm to evolve self-organizing maps from on-line data streams," in *IJCNN (6)*, 2000, pp. 3–8.

[7] V. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics Doklady*, vol. 10, p. 707, 1966.

[8] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, 1970.

[9] M. Fuad and P.-F. Marteau, "The extended edit distance metric," in *Content-Based Multimedia Indexing, 2008. CBMI 2008. International Workshop on*, June 2008, pp. 242–248.

[10] J. B. Kruskal and M. Wish, *Multidimensional Scaling (Quantitative Applications in the Social Sciences)*. SAGE Publications, Inc, 1978.

[11] I. Borg and P. J. F. Groenen, *Modern Multidimensional Scaling: Theory and Applications (Springer Series in Statistics)*. Springer, 2005.

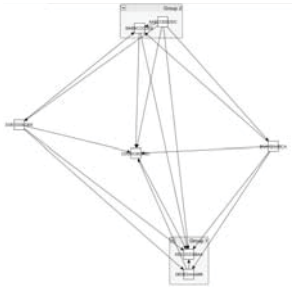

Fig. 8. Toy example 1.

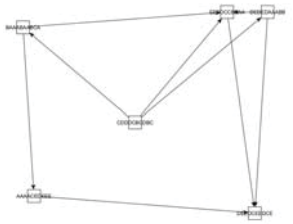

Fig. 9. Toy example 2.

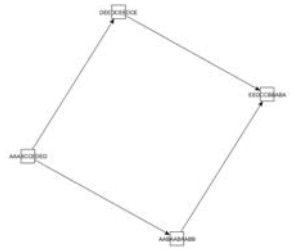

Fig. 10. Toy example 3.

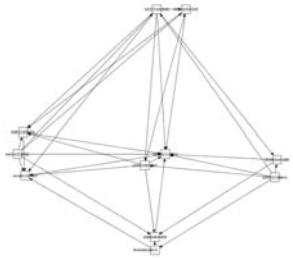

Fig. 11. Toy example 4.

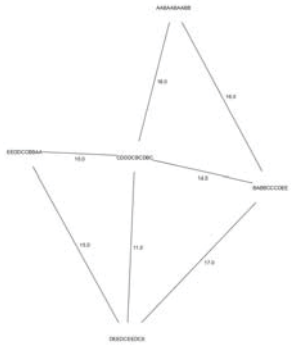

Fig. 12. Toy example 5.

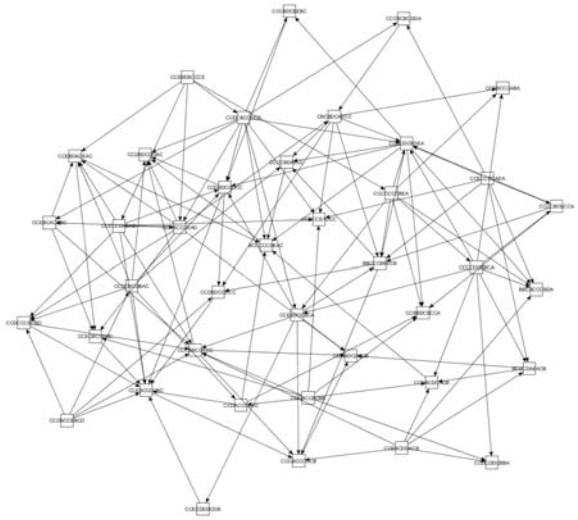

Fig. 13. Layout of the earthquake ESOM.