Real-Time Cooperative Analytics for Ambient Noise Tomography in Sensor Networks

Maria Valero, Fangyu Li, Sili Wang, Fan-Chi Lin, and WenZhan Song, *Senior Member, IEEE*

***Abstract*—The transformative integration of sensor networks and geophysical imaging techniques enables the creation of a system to monitor and analyze seismic data in real time as well as image various subsurface structures, properties, and dynamics. Ambient noise seismic imaging is a technique widely used in geophysical exploration for investigating subsurface structures using recorded background raw ambient noise data. The current state-of-the-art of ambient noise monitoring relies on gathering these high volumes of raw data back to a centralized server or base station to pre-process, cross-correlate, analyze frequency-time components, and generate subsurface tomography. However, modern computational sensors (for example, those with $\sim$1.2 GHz of processor and $\sim$1 GB of memory) can be not only used for recording raw vibration data but also performing *in situ* processing and cooperative computing to generate subsurface imaging in real time. In this paper, we present a distributed solution to apply ambient noise tomography over large dense networks and perform in-network computing on huge seismic samples while avoiding centralized computation and expensive data collection. Results show that our approach can detect subsurface velocity variations in real time while meeting network bandwidth constraints and reducing communication cost ($\sim -75\%$).**

***Index Terms*—Sensor networks, cooperative computing, ambient noise, distributed system, tomography.**

I. Introduction

OVER the last years, ambient noise tomography has become one of the fastest growing research areas in seismology and exploration geophysics. Compared to earthquake-based seismic tomography methods, ambient noise tomography is particularly useful in imaging shallow earth structures [1], [2]. Moreover, because of the persistent nature of the seismic background noise, temporal variation of the earth structure can be analyzed and monitored by studying the variation in the noise cross-correlation function [3]–[5]. Ambient noise methods have the advantage of being low cost and having resistant repeating sources with a minimal environmental disturbance.

Manuscript received March 29, 2018; revised July 23, 2018 and September 5, 2018; accepted October 12, 2018. Date of publication October 17, 2018; date of current version May 8, 2019. This work was supported in part by NSF-CNS1066391, in part by NSF-CNS-0914371, in part by NSF-CPS-1135814, in part by NSF-CDI-1125165, and in part by the Southern Company. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Aleksandar Dogandzic. *(Corresponding author: Maria Valero.)*

M. Valero, F. Li, S. Wang, and W. Z. Song are with the Center for Cyber-Physical Systems, University of Georgia, Athens, GA 30602 USA (e-mail: maria.valero@uga.edu; fangyu.li@uga.edu; sili.wang@uga.edu; wsong@uga.edu).

F.-C. Lin is with the Department of Geology and Geophysics, University of Utah, Salt Lake City, UT 84112 USA (e-mail: fanchi.lin@utah.edu).

Digital Object Identifier 10.1109/TSIPN.2018.2876751

The problem is that the existing ambient noise tomography methods use post-processing approaches to recover subsurface structures, and they do not have the capability of obtaining information in real time. Current approaches involve manual collection of raw seismic data from the sensors to a central server for post-processing and analysis. Sensor network technology has matured to the point where it is now possible to deploy and maintain large networks for earth structures monitoring [6]–[8]. Also, the computing power of every sensor can be one of the most exciting opportunities for in-situ computing due to the ability to generate real-time imaging of the earth's interior and study the complex dynamic processes occurring within. However, it is virtually impossible to collect all raw data to a central place through wireless sensor networks due to the severe energy and bandwidth constraints[1] and disruptions caused by harsh environmental factors. Even though system-level challenges of deploying wireless sensor networks are significant, focusing on distributed in-network signal processing and computing can help to support real-time tomographic imaging.

In this paper, we present a novel real-time ambient noise imaging system through in-situ computing in sensor networks, and we illustrate the process from signal processing challenges to end-to-end system design. The ANSI system is a sensor network of nodes that can efficiently perform seismic ambient noise cross-correlations and compute real-time tomography by continuously monitoring detailed structures within the top few kilometers underground. This system is particularly cost attractive because the ambient noise used for tomographic imaging does not rely on any active sources or earthquakes, and it is autonomous and self-sustainable with all processing and computing in the network. To achieve the goal, we integrate the cutting edge seismic noise analysis, tomography, sensor communication, and large data computation methods. Specifically, we integrate communication and computation devices with sensors such that data recorded by every sensor can be cross-correlated on site with the data recorded at other neighboring sensors, and tomography can be achieved without transmitting the raw data back to a data center. The seismic raw data are also stored at each device database for future analysis; since the sensors form a mesh network, the raw data can be accessed by any device within the network if needed.

The new approach taken in ANSI is general, and it can be implemented as a new field network paradigm for real-time imag-

[1]According to [9], the energy of transmitting 1KB a distance of 100 m is approximately the same that executing 3 million of instructions in one processor. Hence, local data processing is crucial.

ing of highly dynamic and complex environments, including both natural and man-made structures. We believe the system can be applied to a wide range of sustainability-related topics such as hydrothermal [10], volcanic [11], mining safety [12], infrastructure monitoring [13], [14], oil and gas exploration [15], and exploration geophysics [16]–[18]. Additionally, we envision potential future extraterritorial experiments for imaging planetary subsurface structures and activities using ambient noise. In 2018, the *InSight Mission* to Mars [19] is expected to land and begin returning seismic data, which will hopefully greatly improve our knowledge of Mars' interior [20]; in 2020, NASA will also launch *Europa Clipper Mission* that will conduct detailed reconnaissance of Jupiter's moon Europa and investigate whether the icy moon could harbor conditions suitable for life [21]. If we use in-situ computing to process large volumes of network data and only send back continuously updated subsurface images at much lower rates, the required data volume is significantly reduced, which is a necessary step to resolve the issues on the structures and dynamics for extraterritorial bodies. The potential scientific and social impact is significantly and broadly widespread.

The ANSI system proposed here represents a milestone for both earth and computer science efforts. Our approach integrates innovations on ambient noise tomography, in-network computing and signal processing for real-time subsurface imaging as follows: (i) approaches to integrating temporal variation and large N tomography studies based on ambient noise cross-correlation that provide real-time visualization of subsurface as a consequence of geological dynamics and nature resource extraction; (ii) in-network processing techniques to correlate the noise signals between nodes and derive the phase velocity under the limited network resource constraints; and (iii) innovative in-network tomography computing techniques that distribute the tomographic computing burdens to each node while performing real-time seismic imaging generation.

The rest of the paper is organized as follows. Section II presents the related work. Section III provides background information about ambient seismic noise imaging end-to-end process. In Section IV, we present the distributed system design, and the system architecture is explained in Section V. In Section VI, we carry out experiments through real ambient noise data. We discuss results in Section VII. The conclusion and future work are presented in Section VIII.

II. Related Work

Ambient noise seismic imaging has been widely used for extracting surface wave velocity maps in geophysical fields. The method has been applied worldwide (e.g US [22], Asia [23], Europe [24], New Zealand and Australia [25].) A considerable part of these approaches was developed to treat the travel time (time that it takes for seismic waves to travel from one node to another) between every station pair independently. Some approaches, like the one presented by Lin *et al.* [1], have utilized an array of seismic stations to gather information and treat all travel time measurements together to improve the resolution of the tomographic result (velocity map). Even though these approaches have been successfully applied, they lack real-time results. The cross-correlation process needs at least several days for collecting data, and then manual extraction is needed to gather the information to a central server. Thus velocity maps may take days or months in being generated.

Real-time seismic imaging generation is possible across an array of sensors due to the capabilities of current sensor networks. Furthermore, distributed cooperation between nodes for generating seismic imaging has proven to be a milestone in-network computation. Examples of the development of such computing and network methodology for seismic imaging applications can be found in [7], [26]–[29]. All of them have been successful in generating 2D and 3D seismic tomography by applying travel time tomography techniques and using earthquake information to illuminate the subsurface of the earth.

In ambient noise seismic imaging, the data are recovered from ambient seismic noise, which implies no need for active energy sources like earthquakes. Ambient noise imaging can be applied to regions with non-existent seismicity, and it produces reliable measurements at frequencies that are particularly difficult using earthquakes or explosions due to scattering and attenuation. This advantage represents an attractive cheap scenario since producing active energy sources (explosions) in non-seismic areas is very costly.

The first attempt to compute ANSI in distributed sensor networks was made in our previous work [30]. We proposed the use of a method called *distributed eikonal tomography* for generating velocity maps. However, in [30], we assume the travel times have been already calculated to perform eikonal method, and this implies only the last step of the ANSI process. In this paper, we carefully incorporate and improve all previous steps of the ANSI process to get a complete system, namely recording raw data, performing distributed cross-correlation, calculating in-situ frequency-time analysis, deriving travel time measurement, performing distributed eikonal tomography and allowing velocity maps visualization.

To the best of our knowledge, this is the first comprehensive end-to-end system to compute ANSI under distributed constraints through sensor network computing capabilities, cooperation between nodes, and in-situ distributed seismic imaging algorithms. The ANSI prototype system has implemented all steps of ambient noise tomography, from raw data to velocity maps, and it can be extended as a general field instrumentation platform for ambient noise seismic data.

III. Background

Ambient noise seismic imaging is a kind of passive imaging where vibrations of the ambient noise recorded by passive sensor arrays can be used to image the medium through which waves travel. To perform tomography with ambient noise, many methods can be used, for example eikonal tomography [31], straight-ray tomography [32], seismic interferometry [33]. All of them have its own properties and mathematical formulation. For straight-ray and interferometry, an inversion problem needs

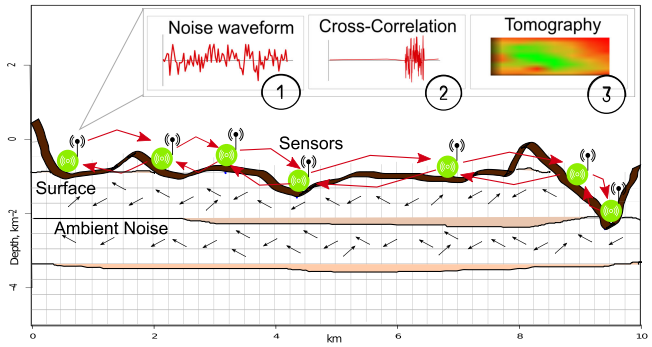
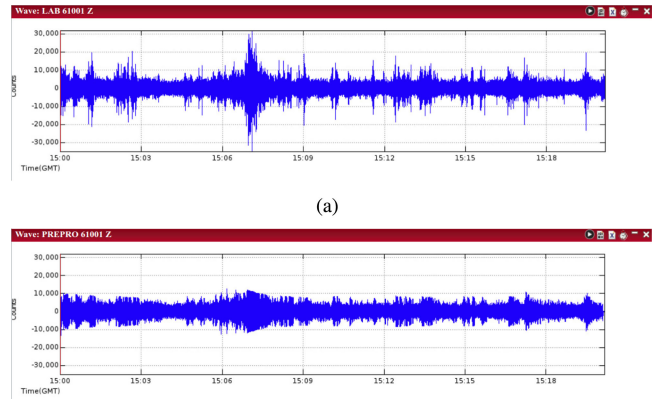


Fig. 1. Ambient noise seismic imaging process.



(a)

(b)

Fig. 2. (a) Raw data. (b) Data after temporal normalization.

to be settled. The eikonal method is a surface wave tomography that complements the traditional methods. The main advantages of this method are: (i) there is no explicit regularization; this makes the method largely free from ad hoc choices; (ii) the method account for bent rays, and ray tracing is not needed. The gradient of the phase front provides information about the local direction of wave travel. In traditional tomography, the use of bent rays would need iteration with ray tracing performed on each iteration; and (iii) **the ray tracing, matrix construction and inversion of the traditional methods are not needed. Those have been replaced by surface fitting, computation of gradients and average.** Therefore, the method is very fast in terms of computational cost and suitable for distributed approaches.

The ANSI methodology, that we introduce in this paper, involves the steps shown in Fig. 1. Those include: (i) using seismic sensors (green circles) to measure the vibration of the ambient noise; (ii) calculating the cross-correlation of the signal waves with neighbors and performing a frequency-time analysis to obtain travel time measurements of the ambient noise signal; and (iii) using eikonal tomography [31] to build velocity maps.

We briefly summarize the steps described in [1], [30], [34], [35] as follows:

A. Signal Pre-Processing

The ambient noise raw data gathered from each individual sensor need to be prepared to get a suitable individual waveform for future cross-correlation. As explained in [34], the purpose of this preparation is to accentuate ambient noise by attempting to remove earthquake signals and instrumental irregularities that tend to hide ambient noise. The signal preparation has three important steps: (i) removing instrumental error response and cutting data; (ii) time-domain normalization and (iii) spectral whitening.

To remove instrumental irregularities, the first step is to remove the mean and the trend of the signal. Then a taper is applied to improve signal properties in the frequency domain [36]. A simple cosine taper filter that applies cosine-shaped attenuation function to specified frequencies at low and high frequencies is applied to remove instrument irregularities. Additionally, the data should be cut into a specific time-window to be analyzed in a window fashion. Data can be cut on one day, some hours, o few minutes. This window of time λ will be used for posterior steps (cross-correlation) and stacked together until complete the total time T of the signal.

The next step is time-domain normalization, also called temporal normalization [34]. The time-domain normalization we use is running-absolute-mean normalization [34]. This method computes the running average of the absolute value of the waveform in a normalization time window of fixed length, and it weights the waveform at the center of the window by the inverse of this average. Given a discrete time-series f, the normalization weight is

$$w_n = \frac{1}{2N+1} \sum_{i=n-N}^{n+N} |f_i|, \tag{1}$$

and the normalized datum is $\tilde{f}_n = f_n / w_n$. The width of the normalization window is $2N+1$, and it is used to determine how much amplitude information is retained. The size of N depends on the half of the maximum period of a bandpass filter. Fig. 2 shows an example of how the preprocessing methodology works.

Finally, a spectral normalization is applied. Spectral normalization seeks to reduce broad imbalances in single-station spectra to aid in the production of a broad-band dispersion measurement [34]. Inversely weighting the complex spectrum by a smoothed version of the amplitude spectrum produces the normalized or whitened spectrum. This process is similar to the temporal normalization but using frequency domain spectrum.

B. Signal Cross-Correlation and Green's Functions

After pre-processing of the raw seismic data, the next step to get travel time between two nodes is to apply cross-correlation and stacking processes. Cross-correlation is a common method to process ambient noise data. The cross-correlation should be symmetric as the positive and negative lag signals are averaged. The result of cross-correlation has a positive correlation of the Green's function, and this contains the information of group velocity and phase velocity at different frequencies [37].

Theoretical work by [1] describes how to estimate the Green's function $G_{AB}(t)$ between nodes A and B using the ambient

noise cross-correlation $C_{AB}(t)$ between them:

$$G_{AB} = -\frac{d}{dt}\left[\frac{C_{AB}(t) + C_{AB}(-t)}{2}\right] \qquad 0 \le t < \infty \quad (2)$$

To obtain the unbiased phase and group velocity measures, the cross-correlation should be transformed to the Green's function using equation (2).

C. Stacking

The stacking process is usually employed to increase the signal-to-noise ratio (SNR) of the signal [38]. In this case, we stack the cross-correlation results every time it is performed (every λ minutes). The stacking means superposition and summation. Before stacking, the cross-correlation is normalized by the maximum amplitude. As a side note, we want to mention that there are usually three channels in seismic sensors that measure vibration on X, Y, Z direction respectively. In this work, we focused on the vertical component Z because our main interest is the Rayleigh waves. In literature, it is well-known that long-range coherent noise can be found on the vertical component [39]–[41]. However, this work can be easily extended to be used with horizontal components too. The important part of working with the horizontal component is the rotation of the data in the radial/transverse coordinates. To make the process almost the same, we can borrow the idea of Lin *et al.* [1] and postpone the component rotation until after the cross-correlation by allowing east and north components temporally normalized together. The distributed sensor network would be the same, only adding an extra-step in the preparation of the data is needed.

D. Frequency-Time Analysis

Frequency-time analysis (FTAN) generates the dispersion curve of the Rayleigh wave phase velocity [34]. A whole FTAN process includes: a series of Gaussian band-pass filters to get Green's function with different central frequencies and transformation processes to get the envelope function and phase function of time series data. With the envelope function and phase function, we can generate a figure called FTAN map with the x-axis as apparent period and the y-axis as group velocity. The local maximum point of this map represents the travel time $t_{\max}$ between two nodes [1]. The value of the phase function at $t_{\max}$ can be used to determine phase velocity.

The summary of the signal processing analysis to obtain the travel time measurements is presented in Fig. 3. Suppose two nodes (A and B) need to correlate their ambient noise signals to obtain the travel time measurement between them. The pre-processing process includes performing a uniform down-sampling of the signal (DS), applying data preparation (Pre) as explained in Section III-A and compressing the signal (Cmp) using a compression library. We use *zlib* data compression algorithm [42] and we achieve a compression rate of ∼50%. If an initial bandpass filter is applied (BP) the compression rate is higher. The BP application is configurable is a configuration file. After the communication of this pre-processed data, we perform the cross-correlation (⊗). Every λ minutes (for our test we select 5 minutes due to experts' recommendations)[2] the process is repeated and the cross-correlations are stacked (S). Then a narrow band-pass filter (NBP) is applied at different frequencies. Frequency-time analysis (FTAN) is then applied and we obtain the travel time measurements at different frequencies.

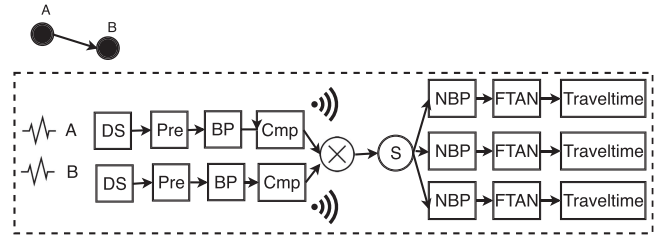


Fig. 3. Summary of signal processing analysis between two nodes for travel time measurements. The acronyms stand for: DS (Down-sampling), Pre (Data preparation), Cmp (Compression), BP (Band-pass filter), ⊗ (Cross-correlation), S (Stacking process), NBP (Narrow Band-pass filter for each frequency in consideration), FTAN (Frequency time analysis), Travel Time (measurements at each particular frequency).

E. Eikonal Tomography

The method of eikonal tomography does not need an initial model of the medium for computing. It only needs the travel times between each pair of stations. The gradient of the travel times provides information about local direction and travel of the wave, hence, deriving phase velocity maps is possible.

1) Eikonal Equation: Once the travel time $\tau(r_i, r)$ are known for positions r (arbitrary point in the medium) relative to a node r_i, the eikonal tomography is performed. The eikonal equation [31] is based on the solution of Helmholtz equation:

$$\frac{1}{c_i(r)^2} = |\nabla\tau(r_i, r)|^2 - \frac{\nabla^2\Lambda_i(r)}{\Lambda_i(r)\omega^2}. \quad (3)$$

At high frequencies, when the second right-hand term is small enough, it can be dropped as:

$$\frac{\hat{k}_i}{c_i(r)} \cong \nabla\tau(r_i, r), \quad (4)$$

where, c_i is the phase velocity for event i at position r. $\hat{k}_i$ is the unit wave direction vector for the event i at position r. ω is the frequency, and Λ is the amplitude of an elastic wave at position r. The gradient is computed relative to the field vector r. Equation (4) is derived from equation (3) by ignoring the second term from the right-hand side. These conclude that the gradient of the travel time is related to the local slowness (1/velocity) at r position, and the direction of propagation of the wave (azimuth) denotes the local direction of the wave. Dropping the second term on the right-hand side of equation 4 is justified when either the frequency is high or the amplitude variation is small [31]. When eikonal tomography is used, there is no need for a tomographic

[2]In literature, different cutting window sizes for cross-correlation has been used; for example, 1-minute window [43] or 30-minute window [44]. We chose 5-min after consulting with Dr. Fan-chi Lin, one of our co-authors, and after doing empirical tests of suitable package size for network transmission. However, this size is configurable in the system.

inversion because taking the gradient of the phase travel time surface gives the local phase speed as a function of the direction of propagation of the wave.

2) Isotropic Wave Speeds: Applying eikonal equation (4) can introduce some errors and usually the phase velocity map is noisy due to imperfections in travel time surface calculation. To overcome this issue, a mean slowness and its standard deviation are calculated in order to obtain the isotropic phase speed.

Traditionally, to compute phase velocity maps through eikonal tomography we need the following: (i) to generate a grid of arbitrary points (r) in the field through interpolation of travel times; (ii) to construct a phase travel time surface for obtaining slowness and azimuth vectors in every effective source relative to each arbitrary point in the grid; (iii) to calculate the mean slowness and standard deviation of the phase travel time surface to overcome errors; and (iv) to invert the final slowness vector to obtain the velocity map.

In the next section, we explain how we design a distributed system for obtaining velocity maps from a series of raw data recording from ambient seismic noise. The centralized approach is also explained to further comparison.

IV. Distributed System Design

During the ANSI computing, two phases need message exchange between nodes. In the first phase, called the **correlation phase**, nodes communicate every λ minutes to cross-correlate its pre-processed data with those from its neighbors; here, there is no need of distributed computation because nodes compute locally their results and only talk to neighbors for sending pre-processed information. In the second phase called the **imaging phase**, nodes calculate its partial maps locally, and then communicate these results to produce the final velocity map; here, a distributed approach to implement in sensor networks is required.

In this section, we provide a detail description of each phase; and particularly for **imaging phase**, we formulate the distributed problem to aggregate the final velocity map. We also compare this distributed approach with a standard centralized solution.

A. Correlation Phase

The overview of the correlation phase is described in Fig. 4. In this phase, every node reads raw data from a medium; for example, a seismic sensor reads seismic waveforms in a field. Once the node has completed λ minutes of the reading process, it activates the next steps: preprocessing, communication, cross-correlation and stacking process. Note that the reading process is continuous and the other processes are done in parallel when they are activated. The pre-processing of the data is made in-situ, and it consists of preparing waveform data from each node individually.

After the preprocessing, the node compresses the data into an UDP (User Datagram Protocol) package[3] and broadcasts the package to its neighbors. The node is also receiving pre-processed data from its neighbors. Notice that this communication process may be asynchronous, and the system is able to handle this situation by using a time-stamp inside the UPD package to let the nodes know which data to correlate. Every node cross-correlates its data with each one of its neighbors and stacks it. The stacking process is referred to add the results up for each λ minutes already processed. Algorithm 1 presents the detail process for correlation phase.

[3]The maximum size of the UDP package is 65KB. However, with the compression technique (using *zlib*) and depending the band-pass filter is applied in the data preparation stage, we achieve a compression rate between 50% and 70% which is significant, and it helps to meet bandwidth constraints.

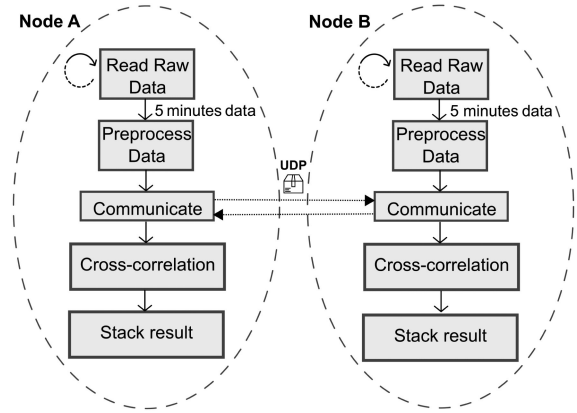


Fig. 4. System design of Phase 1 (correlation phase).

Algorithm 1: Correlation Phase Algorithm.

Define T (Total time for stacking cross-correlation)
Define window size λ
Activate **thread reading** and **thread correlate**
Begin thread reading
while $T - \lambda \geqslant 0$ **do**
  Read data $\mathcal{D}$ from medium;
  if size of $\mathcal{D}$ is corresponding to λ **then**
    Activate **thread prepare**
  end
End thread reading
Begin thread prepare
  Apply down-sampling to $\mathcal{D}$
  Remove instrument noise in $\mathcal{D}$
  Apply Taper process in $\mathcal{D}$
  Apply Frequency whitening process in $\mathcal{D}$
  Compress data $\mathcal{D}$
  Add time-stamp to the compress data $\mathcal{D}$
  Broadcast $\mathcal{D}$
End thread prepare
Begin thread correlate
  Receive data from neighbor i ($\mathcal{D}_i$) and verify time-stamp
  if time stamp in $\mathcal{D}_i$ is equal to time-stamp in $\mathcal{D}$ **then**
    Decompress $\mathcal{D}_i$
    Cross-correlate $\mathcal{D}_i$ and $\mathcal{D}$
    Stack cross-correlation $\mathcal{D}_i$ and $\mathcal{D}$
  end
End thread correlate

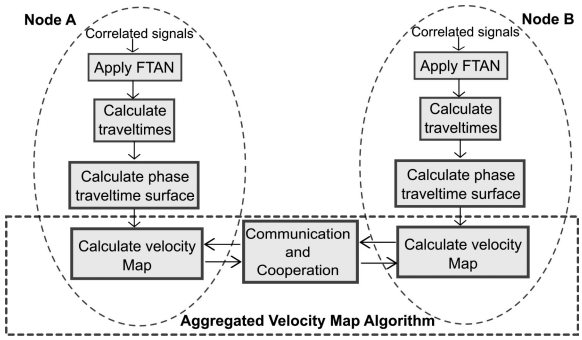


Fig. 5. System design of Phase 2 (imaging phase).

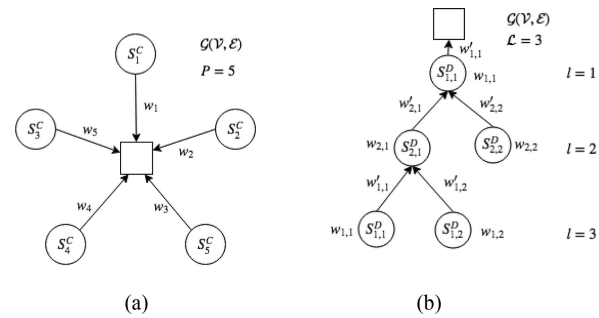


Fig. 6. Topology comparison from centralized imaging and tree-based distributing imaging.

In the algorithm 1, T is the total time we need for stacking results of the cross-correlation. Usually, for getting meaningful velocity maps, we need to stack hours to weeks, depending on the node spacing and noise condition, of cross-correlated data [34]; hence, this parameter is configurable in the system. For our experiments, we stacked one week of cross-correlated data and we got the velocity maps; however, the system correlate in real time, hence, we can generate the velocity map at any moment. λ is the windows size for cross-correlation. We used a window size of five (5) minutes, but this parameter is also configurable in the system. For performing cross-correlation between data of different nodes, we use three different threads. The first thread is called *reading*, and it is responsible for reading data continuously and every λ (5) minutes activating the thread for pre-processing and broadcasting data. The thread for pre-processing and broadcasting data is called *prepare*, and it applies pre-processing techniques to the signal (Section III-A), compresses the data (through compression libraries), creates the package to send, and broadcasts that package to its neighbors. At the same time, the thread *correlate* is listening for receiving packages from neighbor nodes. Once it receives a package, it decompresses the package and verifies the time-stamp to correlate the package data with its own data. Finally, the thread stacks the correlated results for each neighbor. The output of this algorithm is a set of cross-correlated signals between a node and its neighbors.

B. Imaging Phase

The overview of the imaging phase is described in Fig. 5. After completing the correlation phase, every node has a set of correlated signals between its neighbors and itself. The next step is to apply FTAN techniques to obtain travel times (τ) from the cross-correlation results as was explained in Section III-D. Notice that every node calculates individually the travel time between itself and each one of its neighbors.

After travel time calculation, every node constructs its own phase travel-time surface based on its travel time measurements. To construct the phase travel-time surface, the node needs to interpolate its travel time data onto a finer and regular grid. Algorithm 2 describes the process for calculating the phase travel-time surface in each one of the nodes.

Here, every node i executes a interpolation of its travel time measurements in a grid $Gx^\circ \times Gy^\circ$ of size $x \times y$ to get a phase travel-time surface. This grid depends directly on the location of the sensors (nodes) in real field. For example, our simulated study, we used a grid of $1e^{-6^\circ}$ by $1e^{-6^\circ}$ because our real experiments are located in Sweetwater, Texas. For our real deployment, we used a grid of $2e^{-5^\circ}$ by $2e^{-5^\circ}$ because the deployment location only uses ten sensors in a smaller area. However, these parameters are fully configurable in the system. Details of how to fit this grid can be found in [1], [45]. In general, we need to choose an adequate finer, regular grid. The degrees depend on the distance between stations. Larger distances will have higher degrees. The grid also needs the minimum and maximum latitude and longitude to calculate the square regular grid to interpolate.

Algorithm 2: Phase-velocity Travel-time Algorithm (PTT).

1: Input: travel-time measurements τ of node i
2: Interpolate all τ of i onto a $Gx^\circ \times Gy^\circ$ grid size $x \times y$
3: Perform second interpolation of τ with extra tension
4: **for** each point k in the interpolated grid **do**
5: Calculate $\nabla\tau_k$
6: Calculate Slowness S_k
7: Calculate Azimuth A_k
8: **end**
9: Output: S and A vectors for node i

In the next step (line 4), the gradient of each travel-time surface is computed at each spatial node. Using the eikonal equation (equation 4), the magnitude of the gradient allows to calculate the local phase slowness (S), and the direction of the gradient can be used to estimate the azimuth (A).

Once every node completes the calculation of the local phase slowness and azimuth vectors, the second round of communication between nodes begins to calculate a velocity map. Therefore, we need a technique for aggregating partial information of slowness and azimuth inside each node into a final phase velocity map.

There is exists different approaches to aggregate information on sensor networks. The common one is the centralized approach, where all nodes send its data to a central server or SINK. An example is shown in Fig. 6(a). However, the centralized approach introduces a high communication cost in the network, and it is unsuitable for real-time systems. In the distributed approach, Fig. 6(b), an aggregation tree is constructed

for aggregating the partial maps into the final tomography. We are aware of consensus techniques for reaching tomography consensus on sensor networks [30], [46], [47]; however, we chose tree-based aggregation because is faster on real-time systems.

We formally introduce the centralized and distributed approaches in the following way: Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ denote an undirected connected graph (network) with node (sensor) set $\mathcal{V} = (1, ..., P)$ and edge set $\mathcal{E}$, where each edge set $\{i, j\} \in \mathcal{E}$ is unordered pair of distinct node.

1) Centralized Imaging: In the centralized approach, the velocity map $\mathfrak{V}^c$ is calculated based on the slowness observations of $\mathcal{V}$ in the whole network $\mathcal{G}$ as:

$$\frac{\mathbf{1}}{\mathfrak{V}^C} = \tilde{S^C} = \text{argmin} \sum_{i=1}^{P} ||\tilde{S^C} - w_i S_i^C||, \tag{5}$$

where S_i^C is the slowness calculation of node i in a centralized fashion, which means the calculations are already collected in a server. We want to minimize the difference between the estimated final slowness $\tilde{S^C}$ and the aggregated slowness of all nodes $\mathcal{V}$. Note that P is the total number of nodes in the network $\mathcal{G}$. A vector of all ones is defined as $\mathbf{1}$. Because all the slowness measurements are located in a central node, the velocity map $\mathfrak{V}^C$ is reliable and has good resolution. However, the cost of transmitting all measurements to a central place can be significantly high (see Section VI-D.)

2) Distributed Imaging: In the distributed approach, node i carries out communication only with its neighbors $\mathcal{N}_i = (j|\{i, j\}) \in \mathcal{E}$. A randomly formed tree is created to aggregate the velocity map in a bottom top fashion. In this tree-based approach, the aggregation is performed by constructing an aggregation tree, which in this case is formed randomly by pooling neighbor nodes. The flow of data starts from leaves nodes up to the sink and therein the aggregation done by parent nodes.

Let $\mathcal{L}$ be the number of levels of the aggregation tree. Let $S_{l,i}^D$ be the slowness vector or "partial map" of node i in the level l, where $l \in L$ and $i \in \mathcal{V}$ to be aggregated in a distributed fashion. $|S_{l,i}^D|$ is the number of children nodes of node i in the level l.

The final velocity map $\mathfrak{V}^D$ is estimated as:

$$\frac{1}{\mathfrak{V}^D} = \tilde{S}^D = \text{argmin} ||\tilde{S}^D - U_{\text{root}}||, \tag{6}$$

where $U_{root} = U_{1,1}$ is the final slowness vector after the aggregation process at the root node. The process starts from leaves in a bottom-top fashion, and $U_{l,i}$ is calculated for each l level and i node in the level as follows:

$$w'_{l,i} = \begin{cases} w_{l,i} & \text{if } \left|S_{l,i}^D\right| = 0 \\ \sum_{j=1}^{|S_{l,i}^D|} w_{l+1,j} + w_{l,i} & \text{otherwise} \end{cases} \tag{7}$$

$$U_{l,i} = \begin{cases} S_{l,i}^D & \text{if } \left|S_{l,i}^D\right| = 0 \\ \mathcal{Z}_{l,i} & \text{otherwise} \end{cases} \tag{8}$$

$$\mathcal{Z}_{l,i} = \frac{\left(w_{l,i} * S_{l,i}^D\right) + \sum_{j=1}^{|S_{l,i}^D|} w'_{l+1,j} U_{l+1,j}}{w'_{l,i}}, \tag{9}$$

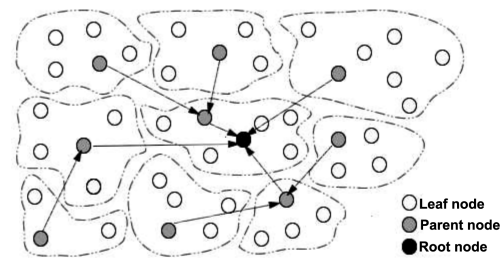


Fig. 7. Example of tree in the aggregation process [30].

where $w_{l,i}$ and $w'_{l,1}$ are the original weight assigned to the partial map and the weight after aggregation respectively. The original weight is assigned in eikonal tomography, and it is related to the azimuth vector [31]. After the aggregation process, the root node contains U_{root} that we can consider the final velocity map.

We design an algorithm to aggregate the information, manage cooperation between nodes, and communicate in a tree structure through broadcasting.

In this approach, we can view the broadcast as a Breadth-First Search (BFS) in the network. Every node i is associated with its level $l(i)$. This level is the length of i shortest path to the root, and it is computed during BFS in the spanning tree. At the beginning, the level of every node i is $l(i) = \infty$, and the level of the root r is $l(r) = 1$. For creating the spanning tree, the r node makes a broadcast. This message contains the root level $l(r) = 0$. When a node i receives a message from a node j contained $l(j)$, i checks its $l(i)$ value to see if $l(i) = \infty$. If this happens, i sets its level to $l(i) = l(j) + 1$ and forward the query to its neighbors; otherwise, i stores the level of j as $l(j)$. The tree has been formed when all l-values are less than ∞. An example of tree is shown if Fig. 7.

The algorithm then computes the aggregation function in a bottom-up fashion in the spanning tree. Every node has to wait until it receives the information from its children, or reaches maximum time. Then, the node aggregates the information and sends it to all of its parents in the tree. In the end, the final velocity map is displayed in the root node.

Algorithm 3 is called *aggregated velocity map algorithm (AVMA)*, and it illustrates the process from a single node point of view. First, each node sets its own level in ∞ to start forming the tree. Then, if the node is selected as root, it sets its $l(i)$ in one. In the emulation scenario, the root node is selected randomly. In a real scenario, the root node is selected during the deployment process. If the node is a root, it waits until receiving the aggregated information from its children (or, alternatively, reaches maximum time) and creates the final phase velocity part by inverting the final slowness vector (line 10). If the node is not a root, it can be either a leaf or a parent node. We know a node is a leaf when it does not have any child. At the same time, each node registers its father in the tree. If the node is a leaf (lines 15-16), it just needs to send its information to its father. If the node is a parent node (lines 17-25), it broadcasts its level $l(i)$ and waits for aggregating the information of its children with its own information. Once the parent node finishes the aggregation process, it sends the results to its father (line 22).

Algorithm 3: Aggregated Velocity Map Algorithm (AVMA).

1: Define i as node number in the network.
2: Initialize node level $l(i) = \infty$
3: **if** i is selected as root **then**
4: Set $l(i) = 0$
5: Broadcast $l(i)$ to its neighbors $|N_i|$
6: **While** until receive from all children or reach MaxTime
7: Receive $(S_j(t), A_j(t), w_j(t))$ from j where $j \in |N_i|$
8: Update $S_i(t+1), A_i(t+1), w_i(t+1) =$ AGG$(S_j(t), A_j(t), w_j(t)$ and $S_i(t), A_i(t), w_i(t))$
9: **end while**
10: Calculate final phase velocity map $\mathfrak{V}^D = \mathbf{1}/S_i(t)$
11: Output $\mathfrak{V}^D$
12: **else**
13: Receive $l(j)$ where $j \in |N_i|$
14: Set own level $l(i) = l(j) + 1$
15: **if** $i = leaf$ **then**
16: Send $(S_i(t), A_i(t), w_i(t))$ to its father
17: **else**
18: Broadcast $l(i)$ to neighbors $|N_i|$
19: **While** until receive from all children or reach MaxTime
20: Receive $(S_i(t), A_i(t), w_i(t))$ from j where $j \in |N_i|$
21: Update $S_i(t+1), A_i(t+1), w_i(t+1) =$ AGG$(S_j(t), A_j(t), w_j(t)$ and $S_i(t), A_i(t), w_i(t))$
22: Send $(S_i(t), A_i(t), w_i(t))$ to its father
23: **end while**
24: **end if**
25: **end if**

The aggregation process (AGG$(S_j(t), A_j(t), w_j(t)$ and $S_i(t), A_i(t), w_i(t))$ defined in lines 8 and 21 is not a trivial addition of values. For performing the aggregation of slowness and azimuth vectors, we use statistical averaging. The statistical average used is a weighted average where the weight is calculated based on azimuth vector collected from node i and j (See Section IV-B2).

V. System Architecture

In this section, we present the architecture behind the distributed system design. Fig. 8 presents the overview of the architecture inside each sensor node. The system has a modular design. Each section is independent and can be monitored through a visualization tool for quality control if needed.

A. Architecture Layers

The first layer shows the storage of the data. Inside each sensor node, a MySQL database has been deployed. The database records the raw stream data (for future uses if needed), the result of cross-correlations, and the generated velocity map. The second layer includes the correlation phase. An important detail is that the node is cross-correlating data every λ time continuously. Even if the nodes are cooperating to estimate the velocity map, the cross-correlation continues. This ensures two important features: (i) the continuity of the system work, and (ii) the possibility of generating velocity maps if a correction in the tomography parameters is needed. To explain better this point, consider the following: the third layer (the imaging phase) is executed after a time T, also defined in the configuration file. Suppose that the system is correlating data every $\lambda = 5$ minutes. After $T = 24$ hours, the imaging phase calculates the velocity map. If the experts realize that they need to do a modification in any of the system parameters (FTAN, tomography, etc.), they can change them and execute again the imaging phase. This introduces flexibility in the system, and it is beneficial for quality control purposes. Furthermore, the modularity feature of our architecture allows to incorporate new algorithms to our system; for example, we can use another type of tomography by disabling the *Eikonal Tomography* module and activating a new tomographic algorithm.

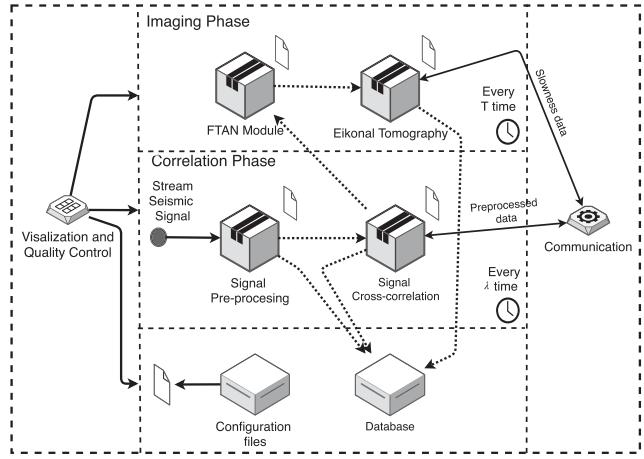


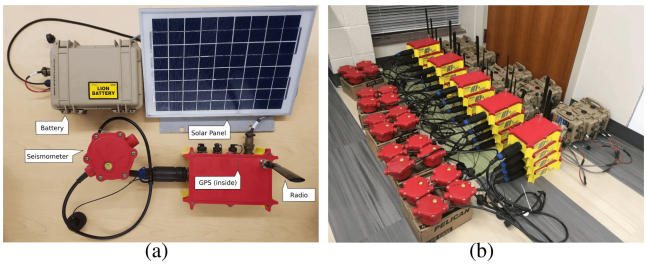
Fig. 8. Node system architecture.

B. Hardware Specification

Our system was tested in two ways: (i) we deployed our system in a network emulator, and we use the seismic data from a previously collected deployment; and (ii) we deployed our system in real devices, and we collected the data directly from the field. The specifications of the emulator and real devices as shown below.

1) Network Emulator: We selected CORE[4] network emulator [48] for validating our system performance. We used CORE emulator because the code developed over it can be easily transferred to a Linux-based device virtually without any modifications. This property is due to the tool allocates for each network node a Linux virtual machine. CORE will allow us to closely emulate the future deployment because we assume the use of Linux-based, tiny but powerful computational units

[4]http://cs.itd.nrl.navy.mil/work/core/



(a) (b)

Fig. 9. (a) R1+ hardware details. (b) R1+ seismograph nodes (for space reasons we omitted solar panels in this picture).

TABLE I
SINGLE-BOARD COMPUTER SPECIFICATIONS

Raspberry Pi 3 Model B	
CPU	1.2GHz 64-bit quad-core ARMv8
Memory	1 GB SDRAM
USB 2.0 ports	4 (via the on-board 5-port USB hub)
On-board storage	32 Gb Micro SDHC
On-board network	10/100 Mbit/s Ethernet, 802.11n wireless, Bluetooth 4.1

(e.g. Beagle-bone Black, Raspberry Pi). Once the system was successfully tested on emulation scenarios, we deployed special sensors on the field for system real-test validation.

2) Field Devices: Every sensor or field device has a global positioning system (GPS), three channel/component seismometer (geophone), a Raspberry Pi 3 board, a battery and a solar panel as shown in Fig. 9. Some hardware components are housed into a waterproof box called R1+ for protecting them from the harsh environment. The low-power GPS interface provides the geo-location of the sensor node and a time-stamp is used for the system to collect, synchronize and process the seismic data. The three channels geophone is incorporated into the system to detect the velocity of ground movements. Each channel records its own data respect to its axis N, E, and Z or directions North, East and Depth (vertical). The single board computer (Raspberry Pi) is the core of the system because is in charge of collecting and storing data, processing data analytics, communicating with other units and providing raw and processed information to a visualization tool. We also integrate a waterproof battery 11 V and 99.9 Wh. The battery is connected to a 10 Watt solar panel for giving to the system renewable energy.

The detailed specifications of the main single-board computer inside R1+ are presented in Table I.

VI. EXPERIMENTS AND SYSTEM EVALUATION

To evaluate our system performance, we conducted two main experiment. The first experiment using CORE emulator, and the second using R1+ devices in the field. Our goal was to that validate our distributed algorithms not only balances the computation load but also achieves low communication cost and high data loss-tolerance.

A. Distributed ANSI Results With CORE Emulator

In this section, we present results of the correlation and imaging phases in our proposed distributed cooperative computing system using CORE emulator. We validated the system design through the use of a real database of ambient seismic noise. We used a time series data recorded by 75 sensors located in the area of Sweetwater, Texas. The data were recorded between March 21 and March 27, 2014.

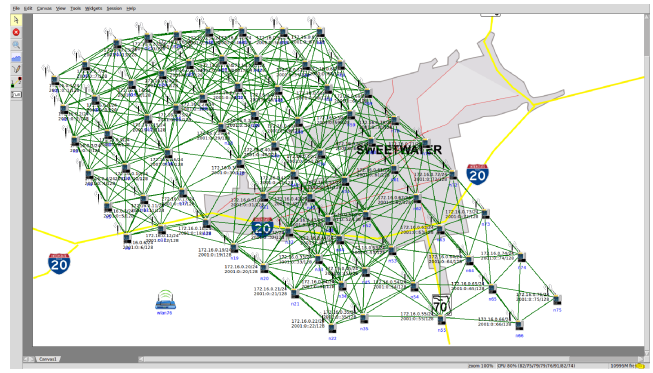
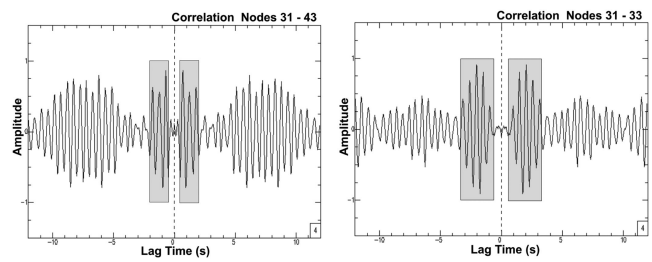


Fig. 10. A deployment of 75 nodes over Sweetwater area in CORE emulator.



(a) (b)

Fig. 11. Cross-correlation results between stations (a) 31 and 43 (1 km distance), (b) 31 and 33 (2.5 km distance). Grey area represents group velocity arrival of the wave signal. The delay time is shorter in (a) as stations 31 and 43 are physically closer than stations 31 and 33.

We deployed the data of each sensor within virtual nodes in the CORE emulator. We carefully designed the deployment structure to match with the physical location of the real sensors. The deployment structure is an important step as cross-correlation of signals is needed. Fig. 10 shows the emulator scenario for Sweetwater database.

As mentioned, the correlation phase is responsible for calculating signal cross-correlation between neighbors. Two examples of our results in the correlation phase are shown in Fig. 11. These results were obtained after exchanging pre-processed data with neighbors every five (5) minutes for seven (7) days. From Fig. 11(a), we can observe the cross-correlation function result between node 31 (red start on Fig. 13(b)) and node 43 (yellow start on Fig. 13(b)) calculated by node 31. These nodes represent the physical sensors 6T497 and 6X497 in the Sweetwater deployment. Fig. 11(b) illustrates another cross-correlation between node 31 and 33 (black start on Fig. 13(b)) calculated by node 31. Node 33 corresponds to the physical sensor 6T536. We configured the system to use a frequency band of 2 Hz.

These cross-correlation functions are used for every node to apply a frequency-time analysis and obtain the estimated travel time between them (Section III-D). When nodes calculate their travel times respective to their neighbors, the imaging phase

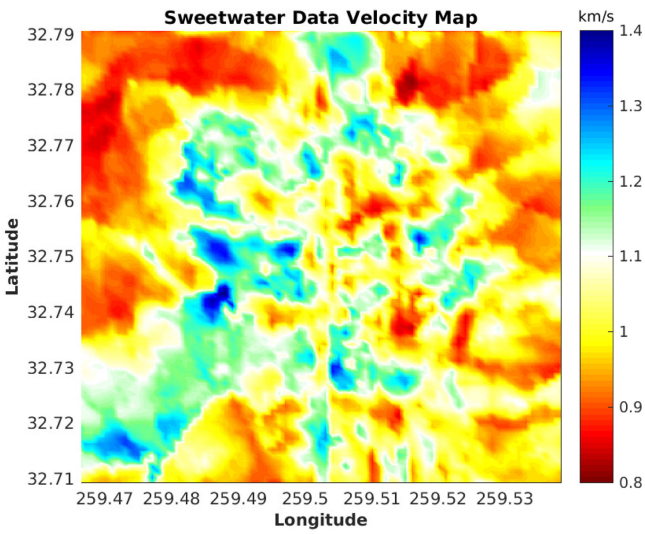
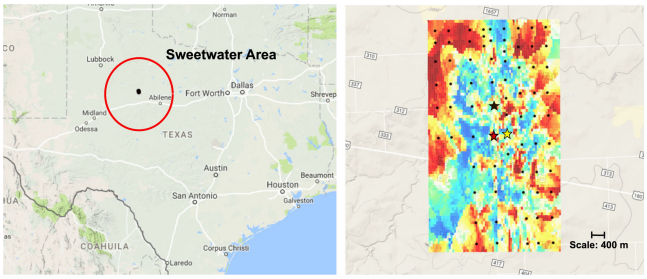


Fig. 12. Velocity map (dominant frequency 2 Hz) for Sweetwater Data using CORE emulator.



(a) (b)

Fig. 13. Location of the analyzed field data and velocity map obtained from the proposed approach. (a) Sweetwater area (red circle) location over Texas. (b) Zoomed in illustration of stations and the velocity map.

begins to calculate the velocity map. Results from the velocity map are illustrated in Fig. 12. We plotted the velocity map over the real location through `Google Maps`. Fig. 13(a) shows the location of the Sweetwater area. Fig. 13(b) illustrates the sensor locations and the final velocity map generated by our system.

B. Distributed ANSI Results With Real Devices

We deployed ten R1+ sensors on the University of Georgia (UGA) campus during January 24th, 2018. The deployment was located in an open area between three main buildings in which there are many pipes under the ground. The Google-Maps location of the devices is shown in Fig. 14. The black box in Fig. 14 shows the ten sensors (nodes) over the field; white box illustrated the location of the nodes respecting each other. We recorded ambient noise data for 7 hours and performed cross-correlation, FTAN and Eikonal Tomography over these data using our system.

From Fig. 15, we can observe the cross-correlation function results between between node 1 (red node on Fig. 14) and node 3 (blue node on Fig. 14). Notice that even though the correlation time was less than 1 day (7 hours), our system was able to obtain identifiable cross-correlation picks that allows FTAN to

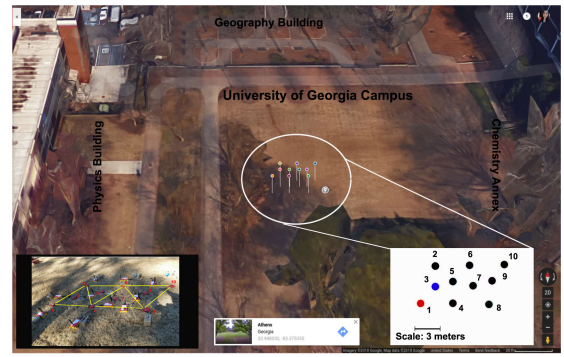


Fig. 14. Deployment of ten R1+ over University of Georgia (UGA) Campus from Google Maps.

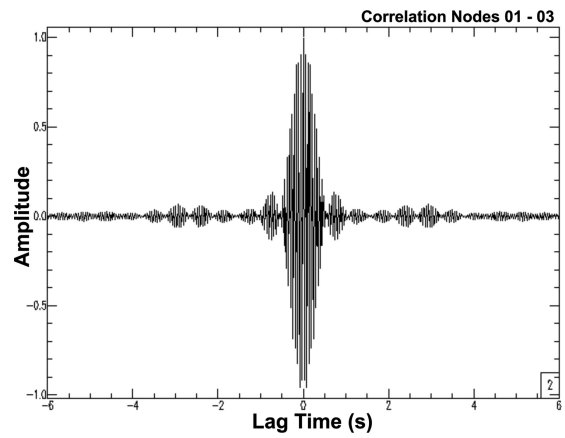


Fig. 15. Cross-correlation result from Station 1 and Station 3 in the real deployment.

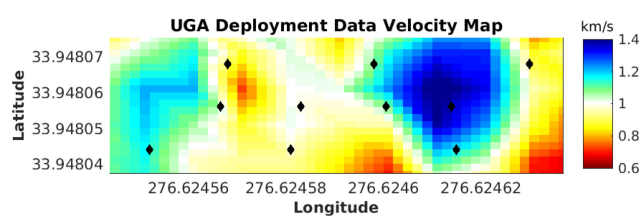


Fig. 16. Velocity map (dominant frequency 35 Hz) obtained from the deployment area at University of Georgia (UGA).

calculate the travel time between the specific two stations. The final velocity map obtained by the Eikonal method is shown in Fig. 16. Black diamonds represent the station locations plotted over the velocity map.

The main idea of this experiment is to test the system functionality and the ability to detect velocity variations using real devices. Because the inter-station distance is small (around 3 meters), we choose a high frequency to be analyzed. The sampling rate of our sensors is 500 Hz. Based on the Nyquist-Shannon sampling theorem, only the first 250 Hz are usable. Furthermore, to avoid aliasing effect [49], only frequencies up to 125 Hz can be adopted. From Fig. 16, in our application, the shallow subsurface velocity is around 1000 m/s. Considering a central frequency of 35 Hz, the wavelength Λ ($\Lambda = c/\omega$, where c is velocity and ω is frequency) will be about 28 m/s. Then,

the seismic resolution is calculated by $\Lambda/4$, resulting in our vertical resolution being about 7 m. Because eikonal tomography is based on computing the spatial gradient of the travel time surface between sensors, the result of this experiment can be unstable due to the number of sensors used. However, the system functionality and the sensors' communication and computation show the possibility of computing ambient noise tomography in networks. If more sensors are added, results of eikonal tomography would be more stable.

C. Robustness Under Unreliable Links

To validate our results, we compare the cross-correlation functions and the velocity map, which was generated by our system with the centralized setup. Using $\tilde{m}$, m^* and $\bar{m}$ to represent the centralized model, the proposed distributed model and the mean value of m^* respectively, we used the following quantitative measures of distance from the centralized model to evaluate the estimation quality

$$e_1 = \left[\sum_{i=1}^{n}(\tilde{m}_i - m_i^*)^2 / \sum_{i=1}^{n}(m_i^* - \bar{m})^2\right]^{1/2}. \quad (10)$$

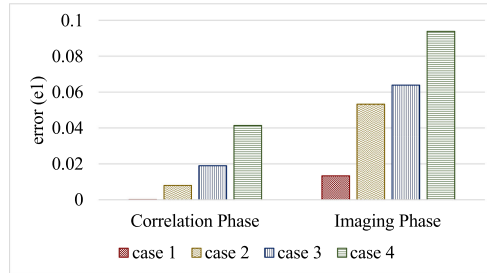
$$e_2 = \sum_{i=1}^{n}|\tilde{m}_i - m_i^*| / \sum_{i=1}^{n}|m_i^*|. \quad (11)$$

These represent the normalized root mean squared distance and the average value distance, respectively.

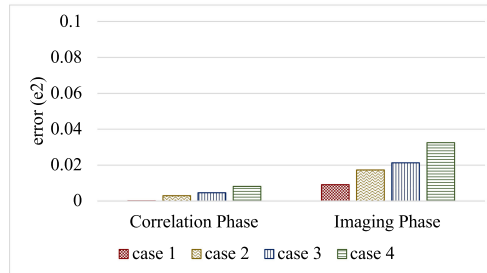
Additionally, one of the important characteristics of our system is the fault tolerance, and here, we validated it by simulating node and link failure. We ran both correlation and imaging phases with four different cases: case 1) No failure case; all nodes communicate correctly and generate cross-correlation functions and aggregate the velocity map; 2) 20% of the nodes fail for 20% of the time; 3) 40% of the nodes fail for 20% of the time; and 4) 60% of the nodes fail for 20% of the time.

First, we described the case 1. Case 1 corresponds with no failure in the system which implies every node correctly computes and communicates all the time. Notice from Fig. 17(a) and Fig. 17(b) that errors e_1 and e_2 for correlation phase are extremely low. This implies that the distributed solution is almost equal to the centralized approach. The same situation can be observed for Case 1 in Imaging phase. Both errors are low because the resultant velocity map is almost identical to the centralized map. Errors are less than 2%.

Since links between node are not always reliable, we design the aforementioned cases 2, 3 and 4. Observe that for correlation phase both errors are very low even when 40% of the nodes fail for 20% of the time. This is due to nodes continuing to correlate every λ (5) minute, and the failures could occur during the inactivity transmission time. Also, the correlation phase is more stable since the stacking process continues stacking results for a long time. However, in the imagining phase, because the aggregation process requires all information of all nodes to be sent to the root node, high failures can significantly increase the error compared to a centralized setup. We plan to overcome this issue by using another technique for combining results in the velocity map such as a consensus algorithm between nodes.



(a)



(b)

Fig. 17. (a) Error e_1 and (b) Error e_2 for the different cases of fault tolerance.

D. Communication Cost

In this section, we present the communication cost of the two main phases of the ANSI system: correlation and imaging. During the correlation phase, the dissemination of the pre-processed data constitutes the major part of communication. Meanwhile, during the imaging phase, the aggregation of local slowness is the process that communicates the most. We evaluated the communication cost of both phases and compared them with a centralized algorithm. In the centralized scheme for the correlation phase, every node sends its corresponding raw data to a base station, or SINK, placed at the center of the array to cross-correlate all data. For the imaging phase, every node sends its slowness calculation to the same SINK station to calculate the final velocity map.

Fig. 18 shows the communication cost in terms of number of received messages for each node during the correlation and imaging phases. Fig. 18(a) and 18(b) correspond to the correlation phase of (a) the centralized approach and (b) our distributed ANSI system respectively. Similarly, Fig. 18(c) and 18(b) correspond to the imaging phase of (a) the centralized approach and (b) our distributed ANSI system.

In the correlation phase with centralized setup, all nodes send the raw data to the SINK every five (5) minutes. Fig. 18(a) shows the number of messages after one (1) hour of communication. Fig. 18(b) presents the communication cost after one (1) hour of communication in the distributed approach; Here, the communication cost is notably less in the whole network as nodes only share information with neighbor nodes. The number of received messages is directly proportional to the number of neighbors. From our deployment structure (Fig. 10), we can see the top

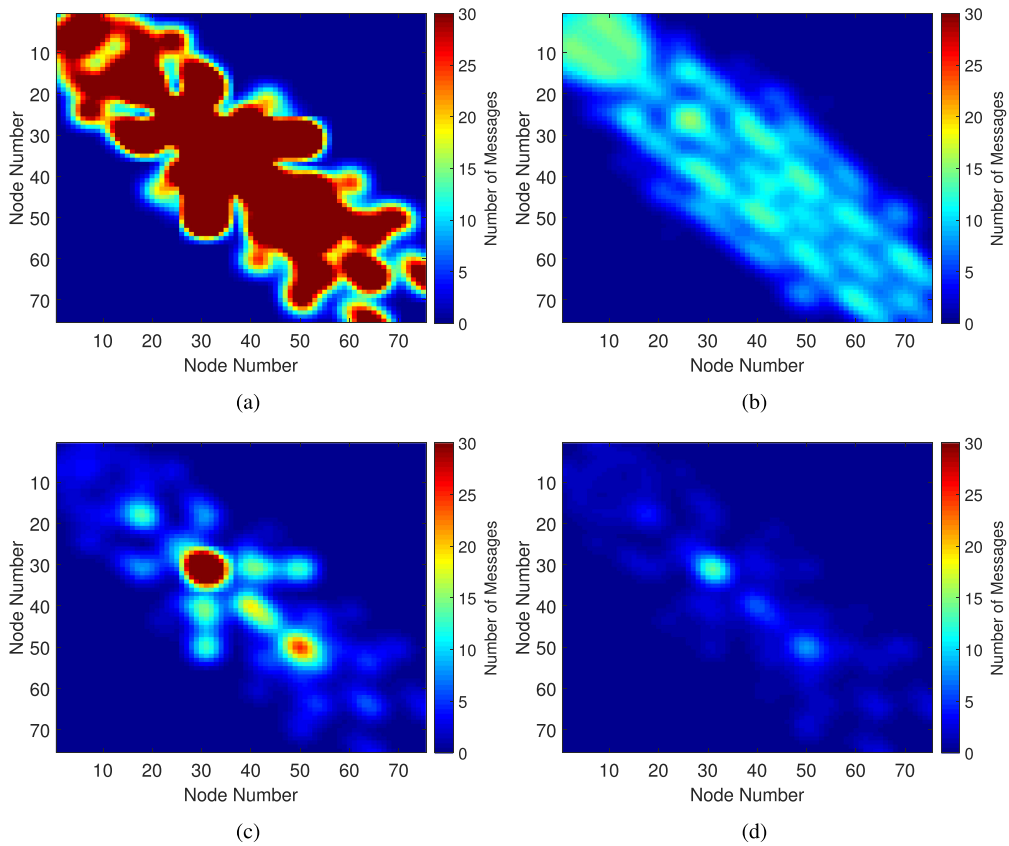


Fig. 18. Communication cost in number of received messages as 2D heat map: (a) Centralized correlation phase, (b) Distributed correlation phase, (c) Centralized imaging phase, and (d) Distributed imaging phase.

left nodes (nodes 1, 2, 3, etc.) have more neighbors than the bottom right nodes (nodes 75, 74, 73, etc); this corresponds to the results obtained in Fig. 18(b) where axis x and y represent the number of nodes in consideration.

In the imaging phase, we measured the communication cost of sending the information of every local slowness to the SINK, and we compared the result with the cost of the distributed approach. Fig. 18(c) illustrates centralized imaging phase. For comparison purposes, we selected as SINK the same node that was selected as root node in the distributed approach. From Fig. 18(d), we can observe the distributed imaging phase communicates fewer messages than the centralized. Notice that the communication cost in distributed imaging phase is higher near and around root node, as the root node has more children than parent nodes.

We also evaluate the communication volume of both phases by measuring the number of megabytes transmitted by every node in the network. Fig. 19 illustrates the communication volume of both phases. In the correlation phase, the communication volume represents the megabytes transmitted over the network. Observe that in the centralized setup the total volume of communication is around 146Mb for completing 1 hour of cross-correlation results. In contrast, the distributed setup transmits around 50Mb for the same hour of cross-correlation results which implies a reduction of approximately 66%. This is basically due to nodes in the distributed approach cooperating to calculate the cross-correlation with only neighbors.

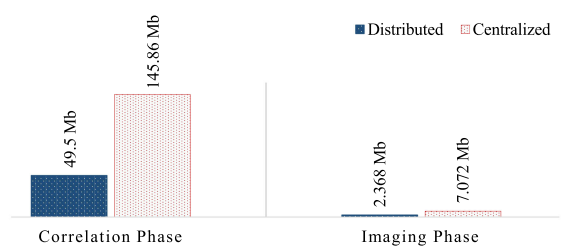


Fig. 19. Communication volume in the network. Correlation phase (after one hour of cross-correlation) and Imaging phase.

The same situation occurs in the imaging phase; the distributed approach reduces by more than 60% the communication volume compared to centralized setup. However, because distributed approach uses broadcasting to communicate with all neighbors, if we measure the communication volume in the network in terms of Mb received, the centralized approach may have equal or less volume than the distributed approach.

E. Computational Cost

We evaluated the computational cost of processing correlation and imaging phases by measuring the CPU times in seconds for each one of the nodes. Fig. 20 illustrates computational cost measurements for centralized and distributed setups imaging

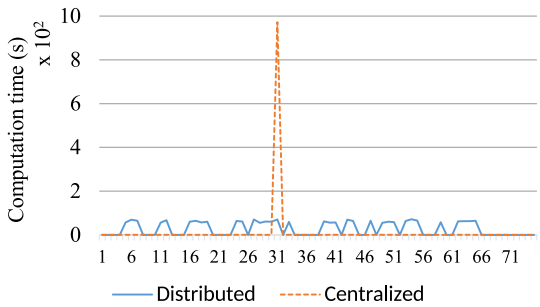


Fig. 20. Computational Cost. X-axis represents number of nodes.

phase. As expected, our distributed system balanced the computational cost as every node calculates its own results and shares them to cooperate in the final result.

F. Effects of Topology

Topology plays a key role in distributed systems like the one presented in this paper. Since the method communicates only with its immediate neighbors, the topology decides how fast the information diffuses in the network. Therefore, the correlation phase on a strongly connected topology generates more information of travel times from immediate neighbors and it may impact the quality of the final velocity map. More neighbors imply more resolution in the velocity maps because there are more calculated travel times to use for interpolating the phase travel-time surface (Sections IV-A and IV-B.).

In traditional ambient noise tomography imaging, the cross-correlation of the signals is performed between all pairs of sensors, which requires $\mathcal{O}(N^2)$ computation. All-to-all cross-correlation represents a large volume of data and the use of nodes as multi-hop nodes in the network. Our system methodology aims to reduce computation and communication complexity by cross-correlating only with neighbor nodes in the mesh network (e.g. K neighboring sensors), which only requires $\mathcal{O}(KN)$ correlations. In strongly connected topologies, the velocity maps result will be almost the same as computing all-to-all cross-correlation. In sparse topologies, when we are only able to observe a subset of entries, the resolution of the recovered tomography may be reduced. However, when the underlying true map varies smoothly (which can be viewed as a low-dimensional structure), the quality of the recovered tomography using partial data will not degrade much from that recovered using the full data. Nonetheless, in our future work, we plan to study some approximation techniques, like matrix completion [50], to recovered cross-correlation from no-neighbor nodes without transferring a large amount of data.

VII. Discussion

In this section, two main aspects of our distributed approach are discussed: (i) the reliability of the eikonal tomography result compared with traditional tomography; and (ii) the trade-off between the centralized and distributed scenario.

The main purpose of our in-field experiment is to test the system ability to detect velocity variation using real devices.

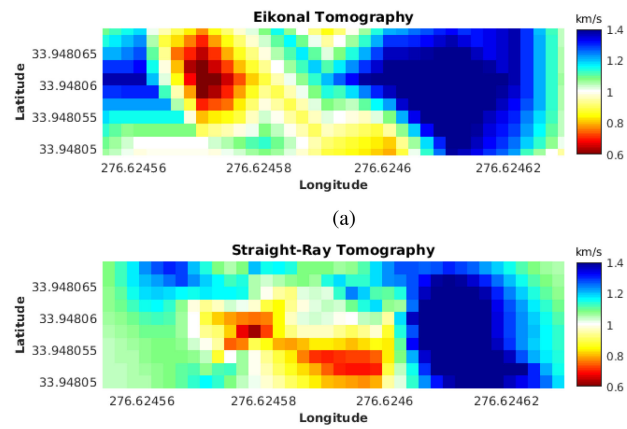


Fig. 21. (a) Eikonal Tomography from UGA deployment with dominant frequency of 35 Hz. (b) Traditional straight-ray tomography of the same deployment at 35 Hz.

As mentioned, due to number of sensors limitations, the experiment was made with ten units only. The results can be unstable due to the short station-separation distance. To measure the resulting stability, we also performed a traditional tomographic method based on straight-ray approximation [32], which can be done only in a centralized fashion, and compared the results. This is a typical comparison to validate ambient noise tomography results [31]. Figs. 21(a) and (b) show the velocity maps obtained by eikonal tomography and straight-ray tomography respectively at a dominant frequency of 35 Hz.

Agreement between the velocity maps produced with eikonal tomography and the traditional straight-ray tomography is generally favorable, but there are some regions with significant disagreements. The main differences likely occur due to the regularization applied in the straight-ray inversion, which tends to distort the velocities near to the edge of the map. This was already reported in [31]. However, from these results, we can see our system is able to recover subsurface velocity differences in a distributed fashion. As mentioned in the system architecture (Section V), other new distributed tomographic methods can be adapted into the system by changing the eikonal tomography module for a new tomographic technique; an adaptation of the input tomographic parameters may be needed too.

Finally, a discussion of the trade-off between the centralized and the distributed approach is presented here. For the image phase, we have made an extensive comparison with the centralized approach in [30]. Notice that because we have used real datasets, there is no ground truth for the velocity of Sweetwater Data and/or UGA deployment. Hence, we focus on the comparison of the proposed method with the centralized processing scheme, which can be used as a benchmark that fully utilizes the data available. Interpretation of this data requires in-depth knowledge of geophysics and is out of the scope of this paper.

Figs. 22(a) and (b) show the final velocity map of the Sweetwater area analyzed in this paper using a centralized approach and distributed approach respectively. There are some disagreement between both maps mainly because the centralized method

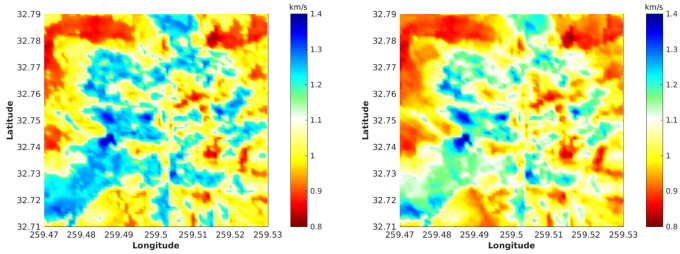


Fig. 22. Velocity map of Sweetwater Database. (a) Centralized approach. (b) Distributed approach.

utilizes more information on cross-correlations. However, the mean squared distance (e_1) and average value distance (e_2) between both approaches are less than 15%, and we can notice a similar pattern in both maps and a differentiation in the structures. This indicates that the distributed method is able to recover similar results than the centralized approach. The main advantage of the distributed method is the communication cost is significantly reduced, and the bandwidth and network constraints are met. The disadvantage relies on the fact of total failure of nodes. As it was explained in Section VI-C, the system is resilient to package lost; however, if a significant portion of the nodes in the network dies, the result will be considerably affected. On the other hand, the centralized approach guarantees more accurate results, but the cost of transferring all data to a central place is very high in terms of sensor networks.

VIII. Conclusion and Future Work

In this paper, we presented an innovative distributed cooperative in-network system for real-time seismic imaging generation through ambient noise data. We integrated in-network processing techniques to correlate the noise signals between sensors and derive the phase velocity under the limited network resource constraints. We showed that computing information at the node level and cooperating with neighbors makes it possible to illuminate near-surface velocities of the earth. We showed that both system phases produce results close to the centralized approach, and they balance communication across the network. Furthermore, we also tested our algorithms under field conditions of sensor networks, such as loss of packages, and showed they are robust in terms of loss tolerance.

We plan to extend this work through the use of other techniques, such as consensus algorithms for combining information in the imaging phase. With the results obtained using real devices, we are also looking forward to focusing on the integration of new algorithms to the distributed ANSI system for more applications like pipe network mapping and leakage detection. Other seismic waves measurements and direct sub-surface modeling can be included in this study.

Acknowledgment

The authors would like to thank the reviewers for their constructive comments to improve this manuscript.

References

[1] F.-C. Lin, M. P. Moschetti, and M. H. Ritzwoller, "Surface wave tomography of the western united states from ambient seismic noise: Rayleigh and love wave phase velocity maps," *Geophys. J. Int.*, vol. 173, no. 1, pp. 281–298, 2008.

[2] M. P. Moschetti, M. H. Ritzwoller, F. Lin, and Y. Yang, "Crustal shear wave velocity structure of the western United States inferred from ambient seismic noise and earthquake data," *J. Geophys. Res.*, vol. 115, Oct. 2010, Art. no. B10306. [Online]. Available: http://dx.doi.org/10.1029/2010JB007448

[3] F. Brenguier *et al.*, "Towards forecasting volcanic eruptions using seismic noise," *Nature Geosci.*, vol. 1, no. 2, pp. 126–130, Jan. 2008. [Online]. Available: http://dx.doi.org/10.1038/ngeo104

[4] F. Brenguier, M. Campillo, C. Hadziioannou, N. M. Shapiro, R. M. Nadeau, and E. Larose, "Postseismic relaxation along the San Andreas fault at parkfield from continuous seismological observations," *Science*, vol. 321, no. Sep., pp. 1478–1481, 2008.

[5] Z. Duputel, V. Ferrazzini, F. Brenguier, N. M. Shapiro, M. Campillo, and A. Nercessian, "Real time monitoring of relative velocity changes using ambient seismic noise at the Piton de la Fournaise volcano (La Réunion) from January 2006 to June 2007," *J. Volcanol. Geothermal Res.*, vol. 184, no. 1/2, pp. 164–173, Jul. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.jvolgeores.2008.11.024

[6] W.-Z. Song, R. Huang, M. Xu, A. Ma, B. Shirazi, and R. Lahusen, "Air-dropped sensor network for real-time high-fidelity volcano monitoring," in *Proc. 7th Ann. Int. Conf. Mobile Syst., Appl. Serv.*, Jun. 2009, pp. 305–318.

[7] L. Shi, W.-Z. Song, M. Xu, Q. Xiao, J. M. Lee, and G. Xing, "Imaging seismic tomography in sensor network," in *Proc. IEEE SECON*, 2013, pp. 327–335.

[8] G. Kamath, L. Shi, and W.-Z. Song, "Component-Average based distributed seismic tomography in sensor networks," in *Proc. 9th IEEE Int. Conf. Distrib. Comput. Sensor Syst.*, May 2013, pp. 88–95. [Online]. Available: http://dx.doi.org/10.1109/DCOSS.2013.17

[9] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Commun. ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[10] R. G. Green, K. F. Priestley, and R. S. White, "Ambient noise tomography reveals upper crustal structure of Icelandic rifts," *Earth Planetary Sci. Lett.*, vol. 466, pp. 20–31, 2017.

[11] F. Brenguier, N. M. Shapiro, M. Campillo, A. Nercessian, and V. Ferrazzini, "3-D surface wave tomography of the Piton de la Fournaise volcano using seismic noise correlations," *Geophys. Res. Lett.*, vol. 34, no. 2, 2007, Art. no. 5.

[12] G. Olivier, F. Brenguier, M. Campillo, R. Lynch, and P. Roux, "Body-wave reconstruction from ambient seismic noise correlations in an underground mine," *Geophysics*, vol. 80, no. 3, pp. KS11–KS25, 2015.

[13] L. Liu, Q.-f. Chen, W. Wang, and E. Rohrbach, "Ambient noise as the new source for urban engineering seismology and earthquake engineering: A case study from Beijing metropolitan area," *Earthquake Sci.*, vol. 27, no. 1, pp. 89–100, 2014.

[14] A. Mordret, H. Sun, G. A. Prieto, M. N. Toksöz, and O. Büyüköztürk, "Continuous monitoring of High-Rise buildings using seismic interferometry," *Bulletin Seismological Soc. Amer.*, vol. 107, no. 6, pp. 2759–2773, 2017.

[15] P. Hanssen, "Passive seismic methods for hydrocarbon exploration," in *Proc. 3rd EAGE Passive Seismic Workshop-Actively Passive*, 2011.

[16] A. Mordret, M. Landès, N. M. Shapiro, S. C. Singh, P. Roux, and O. I. Barkved, "Near-surface study at the Valhall oil field from ambient noise surface wave tomography," *Geophys. J. Int.*, vol. 193, no. 3, pp. 1627–1643, 2013.

[17] S. de Ridder and J. Dellinger, "Ambient seismic noise eikonal tomography for near-surface imaging at Valhall," *Leading Edge*, vol. 30, no. 5, pp. 506–512, 2011.

[18] G. Tomar, E. Stutzmann, A. Mordret, J.-P. Montagner, S. C. Singh, and N. M. Shapiro, "Joint inversion of the first overtone and fundamental mode for deep imaging at the Valhall oil field using ambient noise," *Geophys. J. Int.*, vol. 214, no. 1, pp. 122–132, 2018.

[19] W. B. Banerdt, *et al.*, "InSight: A Discovery mission to explore the interior of Mars," in *Proc. 44th Lunar Planetary Sci. Conf.*, Houston, TX, USA, 2013. [Online]. Available: http://www.lpi.usra.edu/meetings/lpsc2013/pdf/1915.pdf

[20] M. P. Panning *et al.*, "Planned products of the Mars structure service for the InSight mission to Mars," vol. 211, pp. 611–650, 2017. [Online]. Available: http://dx.doi.org/10.1007/s11214-016-0317-5

[21] C. B. Phillips and R. T. Pappalardo, "Europa clipper mission concept: Exploring Jupiter's ocean moon," *Eos, Trans. Amer. Geophys. Union*, vol. 95, no. 20, pp. 165–167, 2014.
[22] G. Ekström, G. A. Abers, and S. C. Webb, "Determination of surface-wave phase velocities across USArray from noise and Aki's spectral formulation," *Geophys. Res. Lett.*, vol. 36, no. 18, 2009, Art. no. L18301.
[23] L. Fang, J. Wu, Z. Ding, and G. F. Panza, "High resolution Rayleigh wave group velocity tomography in North China from ambient seismic noise," *Geophys. J. Int.*, vol. 181, no. 2, pp. 1171–1182, 2010.
[24] A. Villasenor, Y. Yang, M. H. Ritzwoller, and J. Gallart, "Ambient noise surface wave tomography of the Iberian Peninsula: Implications for shallow seismic structure," *Geophys. Res. Lett.*, vol. 34, no. 11, 2007, Art. no. L11302.
[25] P. Arroucau, N. Rawlinson, and M. Sambridge, "New insight into cainozoic sedimentary basins and palaeozoic suture zones in southeast Australia from ambient noise surface wave tomography," *Geophys. Res. Lett.*, vol. 37, no. 7, 2010, Art. no. L07303.
[26] G. Kamath, L. Shi, W.-Z. Song, and J. M. Lees, "Distributed travel-time seismic tomography in Large-Scale sensor networks," *J. Parallel Distrib. Comput.*, vol. 89, pp. 50–64, 2016.
[27] L. Zhao, W.-Z. Song, L. Shi, and X. Ye, "Decentralized seismic tomography computing in Cyber-Physical sensor systems," *Cyber-Phys. Syst., Taylor Francis*, vol. 1, 91–112, 2015.
[28] G. Kamath, L. Shi, E. Chow, and W.-Z. Song, "Distributed tomography with adaptive mesh refinement in sensor networks," *Int. J. Sensor Netw.*, vol. 23, pp. 40–52, 2015.
[29] G. Kamath, W.-Z. Song, P. Ramanan, L. Shi, and J. Yang, "DRISTI: Distributed real-time in-situ seismic tomographic imaging," in *Proc. 14th Int. Conf. Ubiquitous Comput. Commun.*, Liverpool, U.K., 2015, pp. 1244–1251.
[30] M. Valero, J. Clemente, G. Kamath, Y. Xie, F.-C. Lin, and W. Song, "Real-Time ambient noise subsurface imaging in distributed sensor networks," in *Proc. IEEE Int. Conf. Smart Comput.*, 2017, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/SMARTCOMP.2017.7947040
[31] F.-C. Lin, M. H. Ritzwoller, and R. Snieder, "Eikonal tomography: surface wave tomography by phase front tracking across a regional broad-band seismic array," *Geophys. J. Int.*, vol. 177, no. 3, pp. 1091–1110, 2009.
[32] M. P. Barmin, M. H. Ritzwoller, and A. L. Levshin, "A fast and reliable method for surface wave tomography," *Pure Appl. Geophys.*, vol. 158, pp. 1351–1375, 2001.
[33] H. Nicolson, A. Curtis, B. Baptie, and E. Galetti, "Seismic interferometry and ambient noise tomography in the British isles," *Proc. Geol. Assoc.*, vol. 123, no. 1, pp. 74–86, 2012.
[34] G. D. Bensen *et al.*, "Processing seismic ambient noise data to obtain reliable broad-band surface wave dispersion measurements," *Geophys. J. Int.*, vol. 169, no. 3, pp. 1239–1260, 2007.
[35] M. H. Ritzwoller, F.-C. Lin, and W. Shen, "Ambient noise tomography with a large seismic array," *Comptes Rendus Geosci.*, vol. 343, no. 8, pp. 558–570, 2011.
[36] V. Madisetti, *The Digital Signal Processing Handbook*. Boca Raton, FL, USA: CRC Press, 1997.
[37] R. Snieder, "Extracting the green's function from the correlation of coda waves: A derivation based on stationary phase," *Phys. Rev. E*, vol. 69, no. 4, 2004, Art. no. 046610.
[38] S. Grion and A. Mazzotti, "Stacking weights determination by means of SVD and cross-correlation," in *Proc. SEG Tech. Program Expanded Abstr.*, 1998, pp. 1135–1138.
[39] J. Rhie and B. Romanowicz, "Excitation of earth's continuous free oscillations by atmosphere–ocean–seafloor coupling," *Nature*, vol. 431, no. 7008, pp. 552–556, 2004.
[40] L. Stehly, M. Campillo, and N. M. Shapiro, "A study of the seismic noise from its long-range correlation properties," *J. Geophys. Res., Solid Earth*, vol. 111, no. B10, 2006, Art. no. B10306.
[41] Y. Yang and M. H. Ritzwoller, "Characteristics of ambient seismic noise as a source for surface wave tomography," *Geochem., Geophy., Geosys.*, vol. 9, no. 2, 2008.
[42] N. Kimura and S. Latifi, "A survey on data compression in wireless sensor networks," in *Proc. Int. Conf. Inf. Tech., Coding Comput.*, 2005, vol. 2, pp. 8–13.
[43] S.-M. Wu, K. M. Ward, J. Farrell, F.-C. Lin, M. Karplus, and R. B. Smith, "Anatomy of old faithful from subsurface seismic imaging of the yellowstone upper geyser basin," *Geophys. Res. Lett.*, vol. 44, no. 20, pp. 10240–10247, 2017.
[44] K. J. Seats, J. F. Lawrence, and G. A. Prieto, "Improved ambient noise correlation functions using Welch's method," *Geophys. J. Int.*, vol. 188, no. 2, pp. 513–523, 2012.
[45] W. H. F. Smith and P. Wessel, "Gridding with continuous curvature splines in tension," *Geophys.*, vol. 55, no. 3, pp. 293–305, 1990.
[46] L. Zhao, W.-Z. Song, X. Ye, and Y. Gu, "Asynchronous broadcast-based decentralized learning in sensor networks," *Int. J. Parallel, Emergent Distrib. Syst.*, pp. 1–19, 2017.
[47] L. Zhao and W. Song, "Decentralized consensus in distributed networks," *Int. J. Parallel, Emergent Distrib. Syst.*, vol. 33, pp. 550–569, 2018. [Online]. Available: http://dx.doi.org/10.1080/17445760.2016.1233552
[48] J. Ahrenholz, T. Goff, and B. Adamson, "Integration of the CORE and EMANE network emulators," in *Proc. Mil. Commun. Conf.*, 2011, pp. 1870–1875.
[49] Wickert, "Sampling and aliasing," 2011. [Online]. Available: http://www.eas.uccs.edu/ mwickert/ece2610/lecture_notes/ece2610_chap4.pdf
[50] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Found. Comput. Math.*, vol. 9, no. 6, 2009, Art. no. 717.

Maria Valero received the M.S. degree in computer science from the University of Los Andes, Mérida, Venezuela, in 2009. She is currently working toward the Ph.D. degree at the College of Engineering, University of Georgia, Athens, GA, USA. She has also been a Professor with the University of Tachira, San Cristobal, Venezuela, since 2004. Her research interests include distributed computing, signal processing, wireless sensor networks, and cyber-physical systems.

Fangyu Li received the bachelor's degree from Beihang University, Beijing, China, and the master's degree from Tsinghua University, Beijing, China, both in electrical engineering, and the Ph.D. degree in geophysics from the University of Oklahoma, Norman, OK, USA, in 2017. He is a Postdoctoral Associate with the College of Engineering, University of Georgia, Athens, GA, USA. His research interests include signal processing, seismic imaging, geophysical interpretation, machine learning, deep learning, distributed computing, and cyber-physical systems.

Sili Wang received the M.S. degree in geophysics from Peking University, Beijing, China, in 2016. She is currently working toward the Ph.D. degree with the College of Engineering, University of Georgia, Athens, GA, USA. Her research interests include signal processing, seismic imaging, and distributed computing.

Fan-Chi Lin is currently an Assistant Professor with the Department of Geology and Geophysics, University of Utah, Salt Lake City, UT, USA. He is one of a few researchers around the world to establish seismic tomography as a discipline with strong theoretical foundations and practical applications. His research interests are mainly focused on seismic interferometry and seismic tomography.

WenZhan Song received the Ph.D. degree in computer science from Illinois Institute of Technology, Chicago, IL, USA, 2005, the B.S. and M.S. degrees from Nanjing University of Science and Technology, Nanjing, China, 1997 and 1999, respectively. He is a Chair Professor with the School of Electrical and Computer Engineering, University of Georgia, Athens, GA, USA. His research focuses on cyber-physical systems and security and their applications in energy, environment, food and health sectors. He was a recipient of the NSF CAREER Award in 2010.