EFFICIENT AND AUTOMATIC WIRELESS GEOHAZARD MONITORING

by

Marc J. Rubin

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Computer Science).

Golden, Colorado

Date _____

Signed: _____
Marc J. Rubin

Signed: _____
Dr. Tracy Camp
Thesis Advisor

Golden, Colorado

Date _____

Signed: _____
Dr. Randy Haupt
Professor and Head
Department of Electrical Engineering and Computer Science

ABSTRACT

In this dissertation, we present our research contributions geared towards creating an automated and efficient wireless sensor network (WSN) for geohazard monitoring. Specifically, this dissertation addresses three overall technical research problems inherent in implementing and deploying such a WSN, i.e., 1) automated event detection from geophysical data, 2) efficient wireless transmission, and 3) low-cost wireless hardware. In addition, after presenting algorithms, experimentation, and results from these three overall problems, we take a step back and discuss how, when, and why such scientific work matters in a geohazardous risk scenario.[1]

First, in Chapter 2, we discuss automated geohazard event detection within geophysical data. In particular, we present our pattern recognition workflow that can automatically detect snow avalanche events in seismic (geophone sensor) data. This workflow includes customized signal preprocessing for feature extraction, cluster-based stratified sub-sampling for majority class reduction, and experimentation with 12 different machine learning algorithms; results show that a decision stump classifier achieved 99.8% accuracy, 88.8% recall, and 13.2% precision in detecting avalanches within seismic data collected in the mountains above Davos, Switzerland, an improvement on previous work in the field.

To address the second overall research problem (i.e., efficient wireless transmission), we present and evaluate our on-mote compressive sampling algorithm called Randomized Timing Vector (RTV) in Chapter 3 and compare our approach to four other on-mote, lossy compression algorithms in Chapter 4. Results from our work show that our RTV algorithm outperforms current on-mote compressive sampling algorithms and performs comparably to (and in many cases better than) the four state-of-the-art, on-mote lossy compression techniques. The main benefit of RTV is that it can *guarantee* a desired level of compression

[1]This chapter fulfills the Science, Technology, Engineering, and Policy requirement of the SmartGeo research program.

performance (and thus, radio usage and power consumption) without subjugating recovered signal quality. Another benefit of RTV is its simplicity and low computational overhead; by *sampling directly in compressed form*, RTV vastly decreases the amount of memory space and computation time required for on-mote compression.

Third, in Chapter 5, we present and evaluate our custom, low-cost, Arduino-based wireless hardware (i.e., GeoMoteShield) developed for wireless seismic data acquisition. In particular, we first provide details regarding the motivation, design, and implementation of our custom GeoMoteShield and then compare our custom hardware against two much more expensive systems, i.e., a traditional *wired* seismograph and a "from-the-ground-up" wireless mote developed by SmartGeo colleagues. We validate our custom WSN of nine GeoMoteShields using controlled lab tests and then further evaluate the WSN's performance during two seismic field tests, i.e., a "walk-away" test and a seismic refraction survey. Results show that our low-cost, Arduino-based GeoMoteShield performs comparably to a much more expensive *wired* system and a "from the ground up" wireless mote in terms of signal precision, accuracy, and time synchronization.

Finally, in Chapter 6, we provide a broad literature review and discussion of *how*, *when*, and *why* scientific work matters in geohazardous risk scenarios. This work is geared towards scientists conducting research within fields involving geohazard risk assessment and mitigation. In particular, this chapter reviews three topics from Science, Technology, Engineering, and Policy (STEP): 1) risk, scientific uncertainty, and policy, 2) society's perceptions of risk, and 3) the effectiveness of risk communication. Though this chapter is *not* intended to be a comprehensive STEP literature survey, it addresses many pertinent questions and provides guidance to scientists and engineers operating in such fields. In short, this chapter aims to answer three main questions, i.e., 1) "when does scientific work influence policy decisions?", 2) "how does scientific work impact people's perception of risk?", and 3) "how is technical scientific work communicated to the non-scientific community?".

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Φ . . . . . measurement matrix used in compressive sampling

Ψ . . . . . sparsity domain used in compressive sampling

α . . . . . sparsity domain coefficients

σ . . . . . standard deviation

Ω . . . . . Ohm: measurement of resistance

$\widehat{x}$ . . . . . decompressed estimate of original signal

M . . . . . length of encoded, compressed signal

N . . . . . length of original, uncompressed signal

V_t . . . . . timing vector used in Randomized Timing Vector

Φ_b . . . . . binary measurement matrix used in Randomized Timing Vector

x . . . . . original (uncompressed) signal

y . . . . . compressed signal encoding

ACKNOWLEDGMENTS

I thank Dr. Tracy Camp for being a terrific advisor, mentor, and friend. I also thank Dr. Alec van Herwijnen for providing the seismic data used throughout this dissertation and Dr. Michael Wakin for helping me understand how compressive sampling works. This dissertation would not have been possible without the endless love and support of my wife, friends, parents, and pets, who reminded me that there's more to life than research.

CHAPTER 1

INTRODUCTION

In this dissertation, we present contributions that address several challenges in implementing an efficient wireless sensor network capable of automatically detecting geohazard events in geophysical data. The new science and technology discussed, though focused on avalanche detection, can be extrapolated to other domains of geohazard monitoring such as: 1) earth dam or levees failure, 2) rockfall or landslides, and 3) volcanic activity. Before delving into the technical aspects of the proposed research, we first describe the motivation for and specifications of the end goal, i.e., an efficient wireless sensor network that can automatically detect avalanches (or other geohazards).

1.1 Motivation

Snow avalanches pose a significant natural hazard to humans. In terms of human impact, from 1990 to 2012, the United States has averaged 25 avalanche fatalities each year [1, 2]. Even though avalanche terrain is found only in a limited geographical portion (i.e., 17 states in the United States), the death rate is nearly half that of other more common natural disasters. As a comparison, from 2003 to 2012 the United States averaged 109 deaths from hurricanes, 109 from tornadoes, 76 from floods, and 35 from lightning per year [3]. In addition to the human cost, avalanches also affect U.S. highways (e.g., Figure 1.1). For example, in Colorado, 160 known avalanche paths impact roads, causing approximately 6,000 to 9,000 hours of avalanche mitigation and cleanup efforts each year [4, 5]. Furthermore, the Colorado Department of Transportation (CDOT) "regularly monitors and/or controls" 278 avalanche paths and triggered over 700 avalanches in 2010-2011 [5].

Since avalanches pose a significant natural hazard, timely and accurate avalanche forecasts are crucial for the safety of many people, including snow removal workers, highway motorists, ski patrollers, ski resort patrons, and backcountry recreationalists. Unlike weather

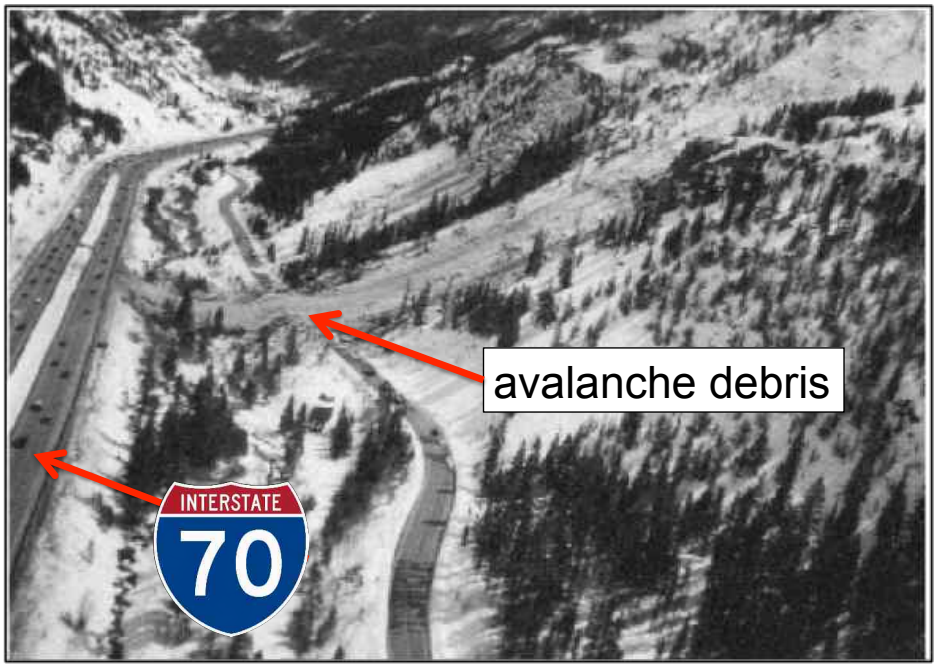


Figure 1.1: Avalanches cause thousands of hours of road closures each season.

predictions, current avalanche forecasts are not based on models of objective, remotely sensed data, but instead rely on the subjective interpretation of weather trends and manual snowpack observations [1, 6]. Avalanche forecasters routinely venture into avalanche terrain to make observations of snow instability, a process that is time-consuming, dangerous, and subjective. Current human-based methods to acquire and analyze avalanche related information can be drastically improved with wireless computing technologies. In particular, avalanche forecasting could make use of wireless sensor network technology and pattern recognition algorithms to obtain spatially precise information regarding avalanche events in near real-time.

For avalanche forecasters, knowing when and where avalanches occur is crucial to making accurate forecasts; avalanche events tell forecasters that the snow is currently unstable and capable of further avalanching [1, 6]. Avalanche forecasting and mitigation institutions constantly monitor "indicator paths", or low-risk avalanche slopes adjacent to more high-risk ones, to help determine when to close highways and set off avalanches with explosives.

Currently, such indicator paths are monitored manually via humans on foot or in vehicles. Such archaic monitoring methodology is time-consuming, spatially sparse, and temporally limited; observers can only venture so far and only during periods of clear visibility. To overcome the inefficiencies associated with manual observations, we envision an efficient wireless sensor network system that can sense and detect avalanche events in near real-time and in any visibility.

1.2 System Overview

To reiterate, timely knowledge of when and where avalanches occur would be invaluable to avalanche forecasters. Thus, we propose a wireless system that can automatically detect avalanches in near real-time (e.g., Figure 1.2). This wireless system will contain three basic elements: 1) sensing, 2) transmission, and 3) detection. Implementing such a novel wireless system requires new research with regard to each component; the sensing interface must be precise yet inexpensive, data transmission must be maximally efficient (while maintaining data quality), and automated avalanche detection must be highly accurate.

Figure 1.2: A cyber-physical system composed of wireless geophones could inform avalanche forecasters that an avalanche occurred.

1.2.1 Precise Wireless Sensing

In the sensing phase, wireless platforms must acquire information from acoustical sensors (e.g., geophones, infrasound microphones) with enough precision to differentiate avalanche and non-avalanche events. Such an approach requires custom electronics; in other words, off the shelf wireless platforms do not meet these high specifications. Additionally, such custom hardware must be inexpensive and easy to use; avalanche forecasting institutions are almost always underfunded and domain experts should not need a computer science or electrical engineering background to work with the devices. To address these requirements, we developed custom, Arduino-based hardware (i.e., a "shield," which we call GeoMoteShield) that is inexpensive, easy-to-use, precise, and accurate. In Chapter 5, we present details regarding the implementation of our GeoMoteShield and, through lab and field testing, compare its performance to a traditional *wired* seismograph and a "from the ground up" wireless mote prototype (i.e., gsMote) developed by SmartGeo colleagues [7]. Results from our work show that our GeoMoteShield performs comparably to the much more expensive, *wired* system and the gsMote prototype WSN. Our contribution is a validated, inexpensive, easy-to-use, Arduino-based "shield" intended for the wireless sensor network, sow science, and geophysics research communities.

1.2.2 Efficient Wireless Transmission

It is well-known in the wireless sensor network community that the radio typically consumes orders of magnitude more power than the other components of a wireless mote (e.g., microprocessor, analog to digital converter)[2] [8]. Thus, reducing radio transmissions is paramount to decreasing the power consumption and overall cost of the wireless system. In other words, less radio usage will equate to a less expensive system; for example, the decreased power requirements for each mote will lead to smaller batteries and less powerful solar panels. This reduction in overall system cost is advantageous because it makes the

[2]In our experience, an XBee Pro 802.15.4 radio consumes about 1250 times more power than all other components on an Arduino Fio mote combined.

wireless system much more appealing to budget-constrained forecasters. In this dissertation, we present our novel on-mote compression algorithm and, using real-world seismic data, compare its performance to other existing techniques. Specifically, in Chapter 3, we present our on-mote compressive sampling[3] (CS) algorithm, Randomized Timing Vector (RTV), and compare it to two existing on-mote CS algorithms in terms of signal recovery and power consumption. In addition, in Chapter 4, we further evaluate our RTV algorithm by comparing it to four state-of-the-art lossy compression techniques found in the literature. Our contributions are twofold: 1) a lightweight, extremely efficient, on-mote compression algorithm (RTV) and 2) a rigorous comparison of RTV against four on-mote lossy compression techniques.

1.2.3 Accurate Avalanche Detection

After precisely sensing and efficiently transmitting the data, the next step is to automatically detect the avalanche events. To be viable, such a detection scheme must be highly accurate and have minimal false positives. In Chapter 2 of this dissertation, we describe our avalanche detection workflow that makes use of signal processing, data mining, and machine learning algorithms to automatically detect avalanches in real-world passive seismic data. As we discuss, solving such a real-world pattern recognition problem is quite difficult. There is often 1) little ground-truth information regarding "true" avalanche events, 2) variability within avalanche event signals, 3) great class imbalance between the avalanche and non-avalanche events, and 4) background noise from spurious seismic events. Thus, solving such a real-world problem requires exploring new tools and techniques to create a robust pattern recognition workflow. Our contribution is a novel pattern recognition workflow to *automatically* detect snow avalanche events in passive seismic data.

[3]More information regarding compressive sampling can be found in Appendices A and B.

1.3 Overview of Dissertation

This dissertation is organized somewhat "backwards", in that we start with automated avalanche detection and work in the reverse logical order (i.e., with transmission second and hardware last). There are two reasons why we selected to present our research results in this reversed order. First, our automated avalanche detection workflow demonstrates the feasibility of detecting geohazard events in seismic data and, therefore, helps motivate the creation of a wireless sensor network. Second, our automated avalanche detection provides additional design requirements for the other chapters; in other words, we use avalanche detection as a metric with which to measure, for example, how well our RTV compression algorithm works.

After describing our technical contributions for 1) automated avalanche detection (Chapter 2), 2) efficient wireless transmission (Chapters 3 and 4), and 3) precise wireless sensing (Chapter 5), we take a step back and consider much broader questions regarding *how*, *when*, and *why* scientific work matters in geohazard risk scenarios. Implementing such a wireless sensor system capable of long-term seismic monitoring has implications beyond the wireless sensor network and snow science research communities; the scientific and technological contributions presented throughout this dissertation could eventually be used in real-world geohazard monitoring applications where people's safety may depend (indirectly) on the results of our scientific work. In Chapter 6, we explore such implications by surveying and analyzing a subset of the Science, Technology, Engineering, and Policy (STEP) literatures to assess how scientific work is: 1) used in policy decisions, 2) used to inform perceptions of risk, and 3) communicated to non-scientific audiences. Chapter 6 is intended to inform scientists and engineers who work in fields of geohazard risk assessment and/or mitigation.

CHAPTER 2

AUTOMATED AVALANCHE DETECTION

A wireless sensor network capable of *automated* continuous geohazard monitoring *must* make use of pattern recognition methods to decrease or eliminate the amount of manual data analysis required. In addition, automation is critical for making such a system near real-time; constant human observation of incoming seismic data is tedious and costly. In this chapter, we address this challenge by describing our novel pattern recognition workflow to automatically detect events in seismic data. Specifically, we discuss how we used signal processing, data mining, and machine learning algorithms to automatically detect avalanches in passive seismic data. Before describing our pattern recognition workflow, as well as experimentation and results, we first motivate our work and summarize previous literature regarding automated avalanche detection.

2.1 Motivation and Background

As discussed in Chapter 1, knowing when and where avalanches occur is critical information to avalanche forecasters and mitigation crews [1, 6]. For example, a sudden increase in natural avalanche activity may lead highway mitigation crews to close sections of highway and artificially trigger avalanches using explosives. Though signs of recent natural avalanche activity is one of the best indicators of impending avalanche danger [1, 6], current observation methods require human interaction, mainly in the form of visual surveillance. During periods of low visibility (e.g., storms, fog, darkness), humans simply cannot view avalanche paths until visibility returns, which may be several hours to many days later. To combat this difficulty, researchers have and continue to use geophones to detect the seismic energy of avalanches rumbling down an avalanche path (e.g., [9–12]). Though many researchers have used seismic sensors to detect avalanches, few have explored the use of machine learning and data mining algorithms to automatically and reliably classify when an avalanche event has

occurred. Following is a summary of the previous literature involving automatic avalanche event detection from seismic data.

The first set of previous research focuses on the SARA (System for Avalanche Recognition Analysis) software suite, which uses an expert system approach based on fuzzy logic rules to detect avalanches from preprocessed seismic signals [13–16]. As detailed in [14], the fuzzy logic rules were not created automatically by any type of rule extraction technique (e.g., RIPPER [17]). Instead, the authors used their "expert knowledge" from the analysis of 308 unambiguously identified seismic events to derive the properties, parameters, and boundaries for each fuzzy logic rule. The rules are mutually exclusive and use features from the time and time-frequency domains to identify the type of seismic events (e.g., helicopter, earthquake, thunder, avalanche, etc.). Though this detection scheme obtained 90% accuracy during testing, the main drawback of SARA is that the fuzzy logic rules are based on manual analysis of previous data; in other words, the rules are not derived automatically or in a timely manner. The authors of [14] acknowledge this drawback, and even state that users must modify the rules to adapt to other sites; however, modifying rules for each site is impractical as it requires significant time and expert knowledge.

In an attempt to automate the training phase of avalanche event classification, [18] used a distanced weighted k-nearest neighbor approach (k = 3) on previously recorded seismic data. Over 10 years of seismic data was recorded from geophones installed in gullies along a dangerous stretch of highway on the coast of Iceland. These geophones recorded many seismic events, including rockfalls, landslides, traffic, and avalanches. The classification results of this automated method were unsatisfactory; specifically, only 65% (78 of 119) of all avalanches were correctly identified using their k-nearest neighbor algorithm approach. This work suggests that there is considerable room for improvement.

To the best of our knowledge, there is no other work that attempts to automatically train classification algorithms to successfully and reliably detect avalanche events from passive seismic data. Our contributions are (1) a thorough description of the preprocessing we do

for feature extraction and (2) a successful pattern recognition workflow to automatically detect avalanches in passive seismic data with 99% overall accuracy and 13.2% precision. We note that precision is not reported in the previous literature discussed.

2.2 Passive Seismic Dataset

During the winter of 2009-2010, seven geophone sensors were buried five to 10m apart in a snow slope in an active avalanche region near Davos, Switzerland (Figure 2.1). The geophones recorded over 100 days of seismic data with 24-bit precision at a sampling rate of 500 Hz.

2.2.1 Raw Data

Our classification experiments deal with this very large data set, i.e., each day contains roughly 43 million 24-bit samples of seismic data and the entire avalanche season is more than four billion 24-bit samples per sensor. More information regarding the deployment can be found in [9].

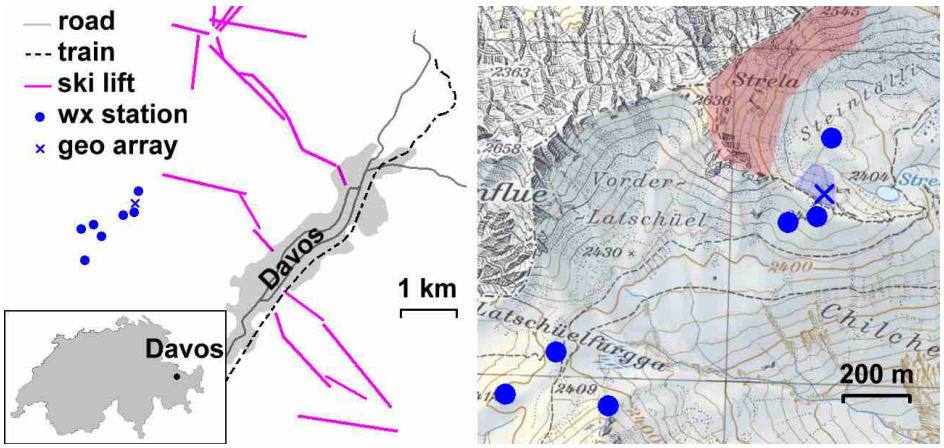


Figure 2.1: The geophone array was installed in a snow slope near Davos, Switzerland. The red shaded portion represents the field of view for two automatic cameras.

2.2.2 Identifying Avalanche Events

In addition to installing geophones, a weather station, acoustic microphone, and automatic camera, which took two images of the adjacent slopes every five minutes (e.g., Figure 2.2), were also deployed. After manually analyzing all consecutive 30 minute spectrograms from a single sensor, camera images, and microphone recordings, 385 potential avalanche events were identified. It is important to note that, despite the microphone and camera systems, there is little ground truth validation of the data set due to poor visibility, lousy microphone data, and camera downtime. Of the 385 potential avalanche events, 33 were confirmed slab avalanches while the remaining 352 were assumed to be small sluff events. Briefly, a sluff is a small surface slide of newly fallen powder snow and is the least dangerous type of avalanche [1, 6]. Slab avalanches, on the other hand, involve large volumes of fast moving, densely packed snow and are known to kill people, destroy structures, and bury roads. It is important to note that our primary goal for this pattern recognition task is to reliably identify the large (confirmed) slab avalanches; correctly identifying the small sluff events is a bonus.

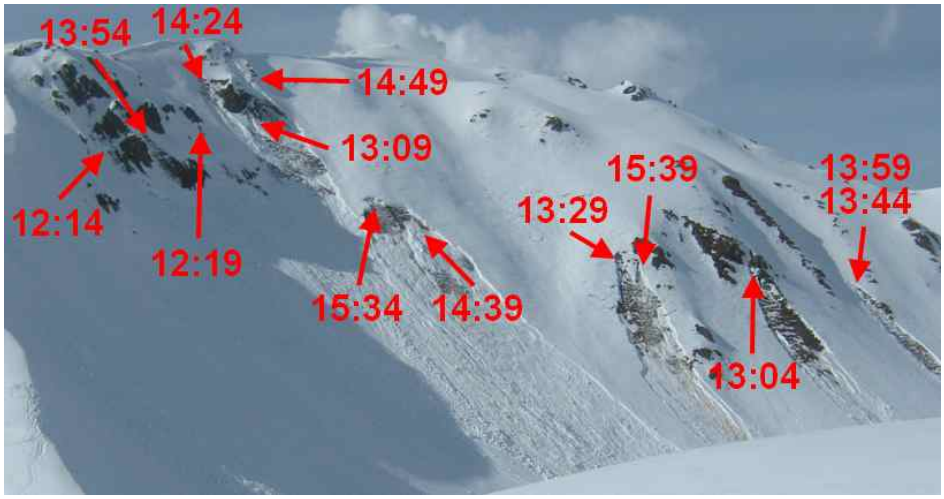


Figure 2.2: Camera images were sometimes used to help confirm the avalanche events (e.g., avalanches from March 23^{rd}, 2011).

Each assumed avalanche event has an estimated start time and length in the passive seismic data. From this data we constructed the classes (i.e., avalanche vs. non-avalanche) for our classification experiments. The avalanches ranged from three seconds to nearly two minutes in length, with a mean length of roughly 25 seconds. We note that the avalanche events only occur approximately 0.2% of the time; in other words, avalanches do not occur approximately 99.8% of the time. We also note that the non-avalanche events are noisy; specifically, there is background and spurious noise caused by wind, helicopters, ski lifts, snow cats, airplanes, and earthquakes (Figure 2.3).

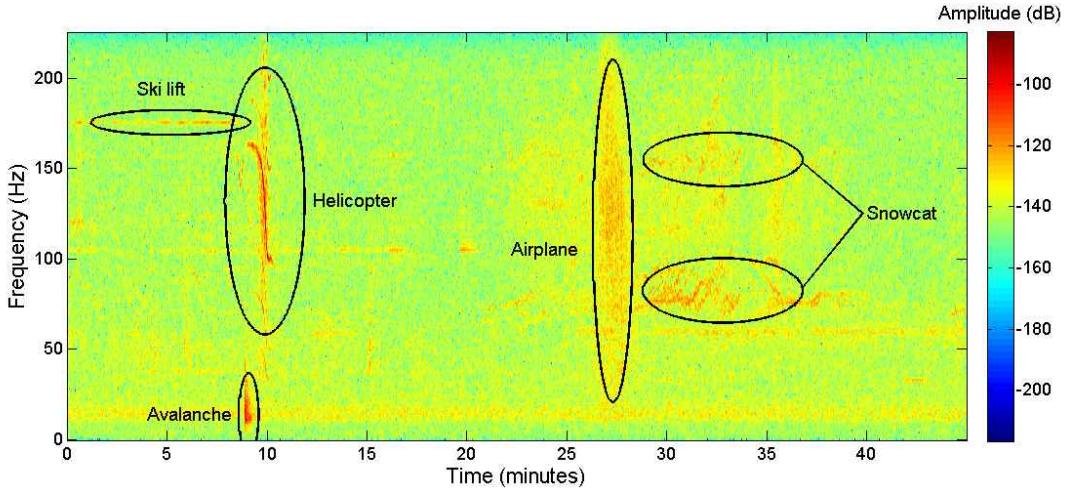


Figure 2.3: Seismic signals generated by various noises in the environment and by a confirmed avalanche.

Though the avalanche events are anomalies in terms of the number of occurrences, we found that statistical and clustering based anomaly detection did not work well with our data set. This result is most likely due to the fact that avalanches were not statistical outliers in the data. We, therefore, experimented with a one-class support vector machine (OCSVM) to detect avalanches, similar to how [19] used an OCSVM to detect anomalous network intrusions. Specifically, we trained and tested an OCSVM using two-fold cross-validation on data from the entire season that was processed using the feature extraction procedure discussed in Section 2.3. Though the results of the OCSVM (Table 2.1) were better than both the statistical and clustering based methods (which both performed no better than

random chance), the classification results for the slab and sluff avalanches were much worse than our pattern recognition workflow discussed next.

In Table 2.1 and the results presented throughout this chapter, we use the following definitions of accuracy, recall, specificity, and precision. First, *accuracy* refers to the percentage of all frames correctly predicted. Second, *recall* is defined as the percentage of avalanche frames correctly predicted. Third, *specificity* is the percentage of non-avalanche frames correctly predicted. Finally, *precision* refers to the percentage of frames labeled as avalanches that were actually avalanches.

Table 2.1: Results from two-fold cross-validation of a one-class support vector machine (OCSVM) trained and tested on all five-second windows within the entire seismic data set.

One-Class Support Vector Machine			
Accuracy	Recall	Specificity	Precision
0.661	0.649	0.661	0.004

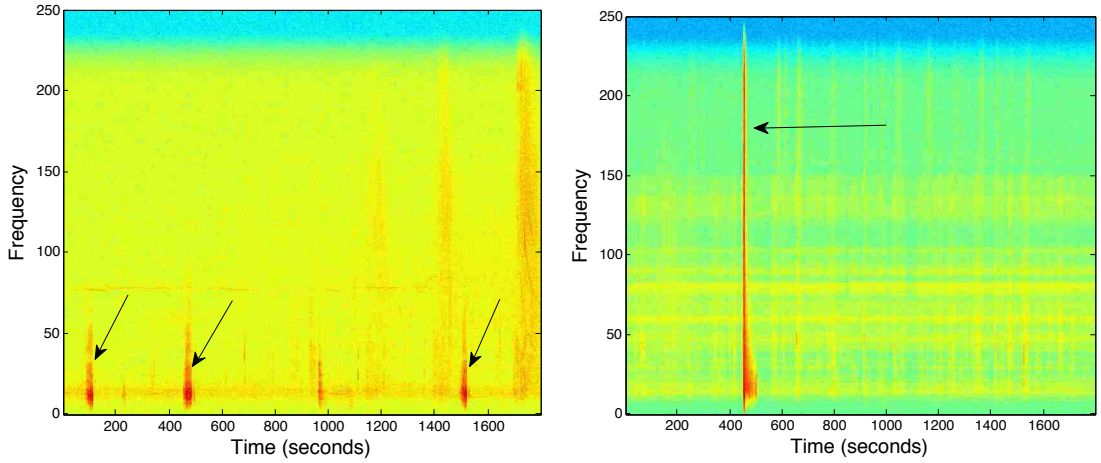
2.3 Feature Extraction

In data mining and machine learning research, the majority of time is spent determining the best features to extract to maximize class separability. This section specifies how we processed the raw seismic data into individual frames containing 10 features appropriate for classification.

2.3.1 Avalanches in the Frequency Domain

A windowed 2048-bin fast Fourier transform (FFT) was used to convert the data from the time to frequency domain. Based on our observations and previous research, avalanches tend to have high energy in the low frequency bins of the spectral domain (Figure 2.4(a)). Unfortunately, increased spectral power in low frequency bins did not always equate to an avalanche, as other non-events (e.g., earthquakes, wind) had significant low frequency amplitudes as well. Additionally, many avalanche events had significant high frequency content and wide spectral traces (Figure 2.4(b)). Needless to say, the ambiguity and variability of

our dataset leads naturally to using automated machine learning algorithms to generate a pattern recognition model.



(a) (b)

Figure 2.4: (a) Avalanches tend to have a strong presence in the low frequency bins of the spectral domain. (b) Some avalanche events had significant energy in high frequency bins as well.

2.3.2 Signal Processing and Feature Generation

We extracted 10 features from the frequency domain using an open-source Matlab toolbox (i.e., MIRToolBox [20]) designed for audio signal processing. Each frame was labeled as either an avalanche or non-avalanche, depending on whether the frame included a suspected avalanche event. Each frame was transformed from the time to frequency domain using a 2048-bin, non-overlapping, windowed FFT. Similar to signal processing techniques commonly found in music signal processing, several spectral features were calculated for each frame (Table 2.2), including the centroid, 85% rolloff, kurtosis, spread, skewness, regularity, and flatness [21].

The seven spectral frame features attempt to create a quantifiable summary of the entire spectral frame. The spectral centroid represents the "center of gravity" of the spectrum, and is a metric often used in music signal processing to measure timbre [21]. 85% rolloff

Table 2.2: The features we extracted all came from the frequency domain. After calculating the spectral features, the frequency spectrum was sorted based on amplitude and the top 1% most energetic frequency bins were used.

Source	**Features**
Spectral Frame	centroid, 85% rolloff, kurtosis, spread, skewness, regularity, flatness
Top 1% Energy Bins	maximum, mean, standard deviation

is the frequency at which 85% of all spectral energy is below. Spectral kurtosis is often described as the "peakiness" of the spectrum, with higher kurtosis meaning more jagged peaks. Spectral spread is simply the standard deviation of the frequency domain. Skewness measures the symmetry of the spectrum. Regularity measures the variability between peaks. Lastly, spectral flatness is a measure of how "flat" the data is, and is calculated as a ratio between the geometric and arithmetic mean.

In addition to the seven features calculated from the entire spectral frame, we sorted the spectral powers based on amplitude and selected the top 1% most energetic frequency bins for further processing. From these top 1% most powerful frequencies, we calculated the maximum, mean, and standard deviation (Figure 2.5).

We note that our experimentation with time domain features (e.g., ratio of short-term over long-term average amplitude) proved fruitless, and will not be discussed further. In addition, because the geophone sensors were buried in the snow (a medium that is notoriously heterogenous), we could not reliably calculate information about the seismic waveforms (e.g., arrival times, velocity, and back azimuth); the seismic velocities between sensors was too highly variably to be useful.

2.4 Avalanche Pattern Recognition

In this section we describe our workflow to detect avalanches from passive seismic data. This section is divided into four parts. First we describe cluster-based subsampling that we

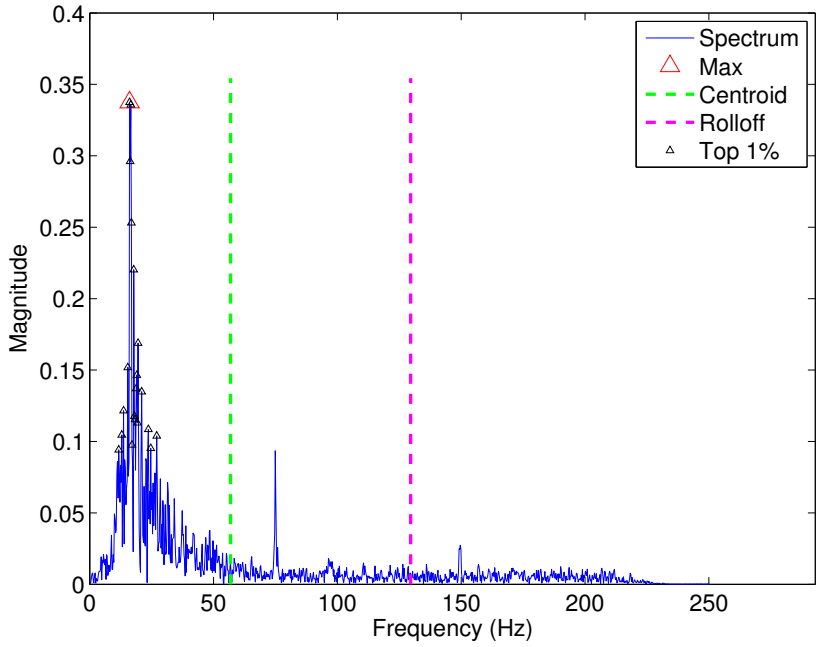


Figure 2.5: For each five second frame, we calculated several features from the frequency spectrum (2048-bin FFT). This event is a confirmed slab avalanche recorded on January 16^{th}, 2011.

used to help offset the deleterious effects of class imbalance. Second, we provide details of our experimentation with 12 different machine learning algorithms on seismic data from the entire season. Third, we test a voting based paradigm that combined information from all seven sensors. Finally, we explore using spectral-flux based event selection to decrease the problem space and increase the overall performance of our classifier.

2.4.1 Cluster Based Subsampling

Before we could begin training and testing, we first addressed the issue of extreme class imbalance. In our entire data set, of the 1,264,648 total 5 second frames, only 2,293 frames contain possible avalanches. Thus, our class distributions were 99.82% non events and 0.18% avalanche events. Classifiers trained and tested according to these imbalanced class distributions could naively obtain over 99% overall accuracy by always selecting the non-event class [22–24]. In such an extreme case, the classifier will be overtrained to recognize the majority

class and will be unable to detect the minority class.

To overcome the issue of class imbalance, we employed a clustering based subsampling methodology (e.g., Figure 2.6). More specifically, we used K-Means clustering to organize the non-avalanche event data into seven different groups. The number of groups was based on the number of known noise events, i.e., planes, helicopters, ski lifts, snow cats, wind, earthquakes, and miscellaneous. Before each iteration of the classifier training and testing, we used stratified cluster-based subsampling to help generate the training and testing datasets. Compared to random subsampling, this method insured that our subsampled data proportionally represented the majority class accurately [17, 24]. In other words, the number of subsampled items from each cluster is proportional to the total number of items within that cluster; for example, in Figure 2.6, one yellow item was subsampled out of seven, two green circles were subsampled out of fourteen, etc.

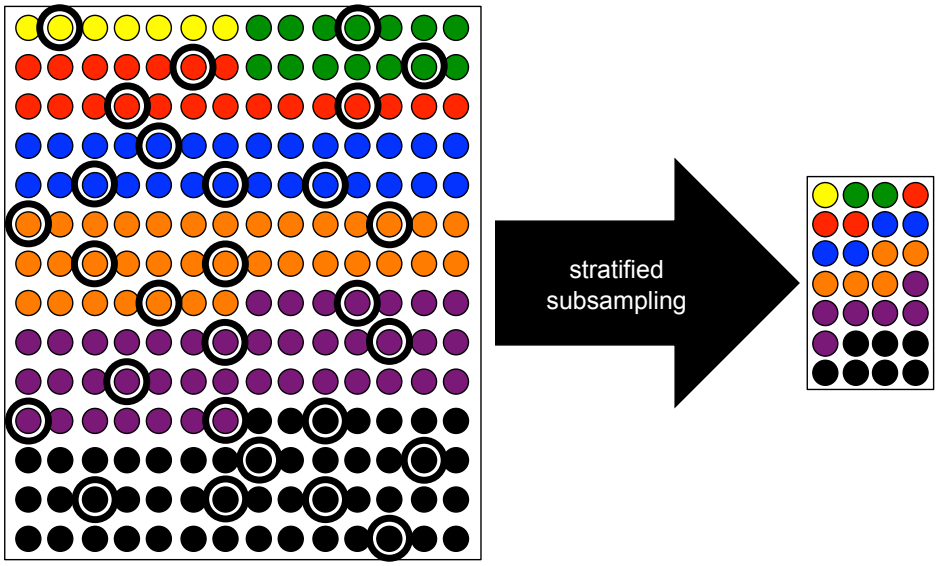


Figure 2.6: Conceptual diagram of cluster-based stratified subsampling. The colors represent possible groups formed during K-means clustering. Of the seven groups in this example, the data is subsampled using stratified (proportional) subsampling, which helps guarantee that the subsampled data accurately represents the majority class.

2.4.2 Full Season Avalanche Classification

We trained and tested 12 different classification algorithms on seismic data from the entire season. More specifically, we trained each classifier on a subset of the passive seismic data from a single sensor and tested on the entire season (excluding the data used for training). The training data consisted of 10% of the 2293 avalanche frames (randomly selected) and an equal number of non-avalanche frames selected via the stratified cluster based subsampling method described previously. This process was simulated 100 times for each algorithm. Training and testing was performed using a combination of open-source data mining tools (i.e., KNIME [25] and WEKA [26, 27]).

For the classification experiments, we tested 12 different classifiers: i.e., artificial neural network (ANN), naive Bayes, Bayes network, CART tree, fuzzy logic rules, Gaussian processes (GaussProc), J48 Tree, k-nearest neighbors (KNN), random forest (RandForest), RIPPER, decision stump, and support vector machine (SVM). This variety was chosen because they represent a wide spectrum of classification algorithms, from probabilistic and statistical (i.e., Bayes and Gaussian processes) to highly non-linear (i.e., ANN and SVM). Furthermore, rule-based classifiers (i.e., fuzzy logic and RIPPER) were used to compare our results with the previous machine learning work done by [13–16]. We chose to use a KNN (k=7) because it represents a simple distance based classifier and was the algorithm of choice in [18]. Lastly, several types of decision trees were tested (i.e., stump, CART, J48, and random forest) because the model's decision boundaries can be interpreted by humans.

The results are quite promising (Table 2.3), i.e., all algorithms reached 84% mean overall classification accuracy or above. Additionally, all classifiers reported mean recall above 77%, meaning that the classifiers could discern the avalanche events.

Though these results show that the dataset is indeed classifiable, the low precision leaves room for improvement. To reiterate, a classifier with 1% precision will identify 99 false positive avalanches for every one true positive avalanche. We note that neither [14] nor [18] report the precision of their classification models. In Section 2.4.4, we investigate ways to

improve precision by using changes in spectral flux to select events before classification.

Table 2.3: Mean results from testing 100 iterations of 12 machine learning algorithms on a single sensor's data. Each algorithm was trained on 10% of the avalanches and an equal number of non-avalanche events selected using cluster-based subsampling. Testing occurred on all data not used for training.

Algorithm	Accuracy	Recall	Specificity	Precision
ANN	0.879	0.846	0.879	0.012
Bayes	0.875	0.831	0.875	0.011
BayesNet	0.944	0.779	0.944	0.023
CART	0.914	0.809	0.914	0.017
Fuzzy	0.842	0.857	0.842	0.009
GaussProc	0.922	0.827	0.922	0.017
J48	0.883	0.842	0.883	0.013
KNN	0.870	0.791	0.870	0.010
RandForest	0.943	0.818	0.943	0.023
RIPPER	0.924	0.812	0.924	0.019
STUMP	0.951	0.770	0.951	0.028
SVM	0.913	0.833	0.914	0.016

We also experimented with a voting based paradigm, where information from all seven sensors was shared to inform classification. In other words, independent classifiers were trained on each sensor, and voting was used during testing to inform the class label. We describe this paradigm next. In Section 2.4.3, we use the random forest classifier algorithm because of its performance in this section.

2.4.3 Voting Based Avalanche Detection

Since data was collected concurrently from seven adjacent geophone sensors, we tested a voting based classification paradigm to increase the precision of our model. That is, we trained random forest classifiers on each of the seven sensors (using the same methodology detailed previously) and, for each five-second frame, tallied the number of avalanche labels. Thus, each frame had zero to seven potential "avalanche" votes. For example, if sensor2 and sensor5 identified an avalanche in frame 17, there would be two votes for classifying an avalanche in frame 17. We selected to use a random forest classifier because such models

are often regarded as being robust against noise; unlike a simplistic decision stump classifier, a random forest classifier is an ensemble learning method that relies on multiple decision tree classifiers simultaneously [17]. In addition, random forest classifiers were one of the top performers in our previous work (see Table 2.3).

We evaluated thresholds for avalanche detection from one to seven votes, and our results are presented in Table 2.4 and Figure 2.7. With increasing votes, our combined classifier approach obtained 99.7% overall accuracy and 13.4% precision, which are better results than single sensor classification (i.e., Table 2.3). At the same time, however, increasing votes led to decreasing recall (true positive rate). Such poor classifier recall results (i.e., 48.7%) led us to further experiment with signal preprocessing, as detailed in the next section.

Table 2.4: Results from testing the voting based classification scheme over the entire season.

Voting with Random Forest Classifiers				
Votes	Accuracy	Recall	Specificity	Precision
1	0.625	0.949	0.624	0.002
2	0.851	0.888	0.850	0.005
3	0.933	0.815	0.933	0.011
4	0.967	0.740	0.967	0.020
5	0.984	0.674	0.984	0.037
6	0.992	0.601	0.994	0.071
7	0.997	0.487	0.997	0.134

2.4.4 Automated Event Selection and Detection

To improve the overall performance of our classification model, we used a more intelligent way to select events before classification. Instead of naively training and testing on *all* data from the entire season, we used spectral flux based event selection to extract only the events with *instantaneous* increases in spectral energy. This technique is often used in music signal processing to extract beat patterns for automated transcription [21]. We apply this technique here, as many non-avalanche events (e.g., helicopter) will not have an instantaneous increase in spectral energy; instead the energy increase will be more gradual.

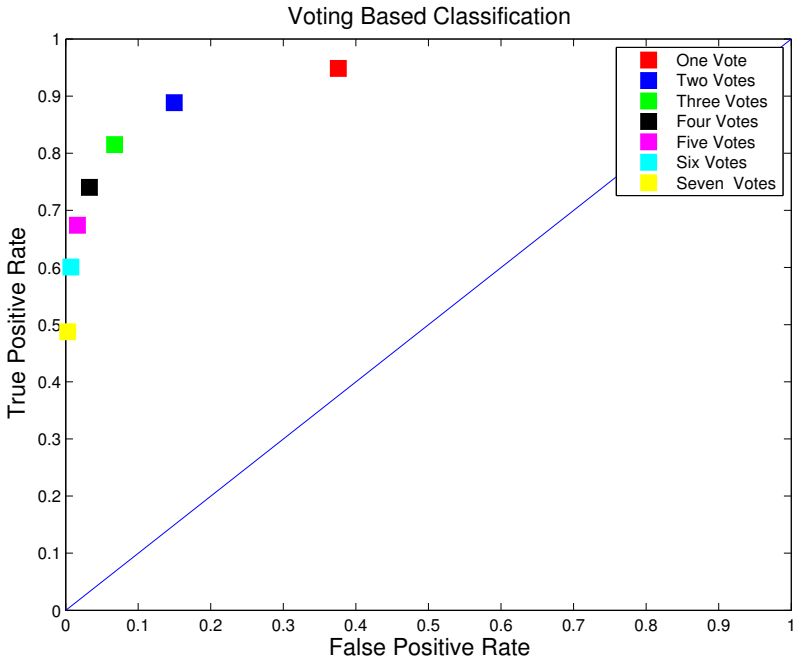
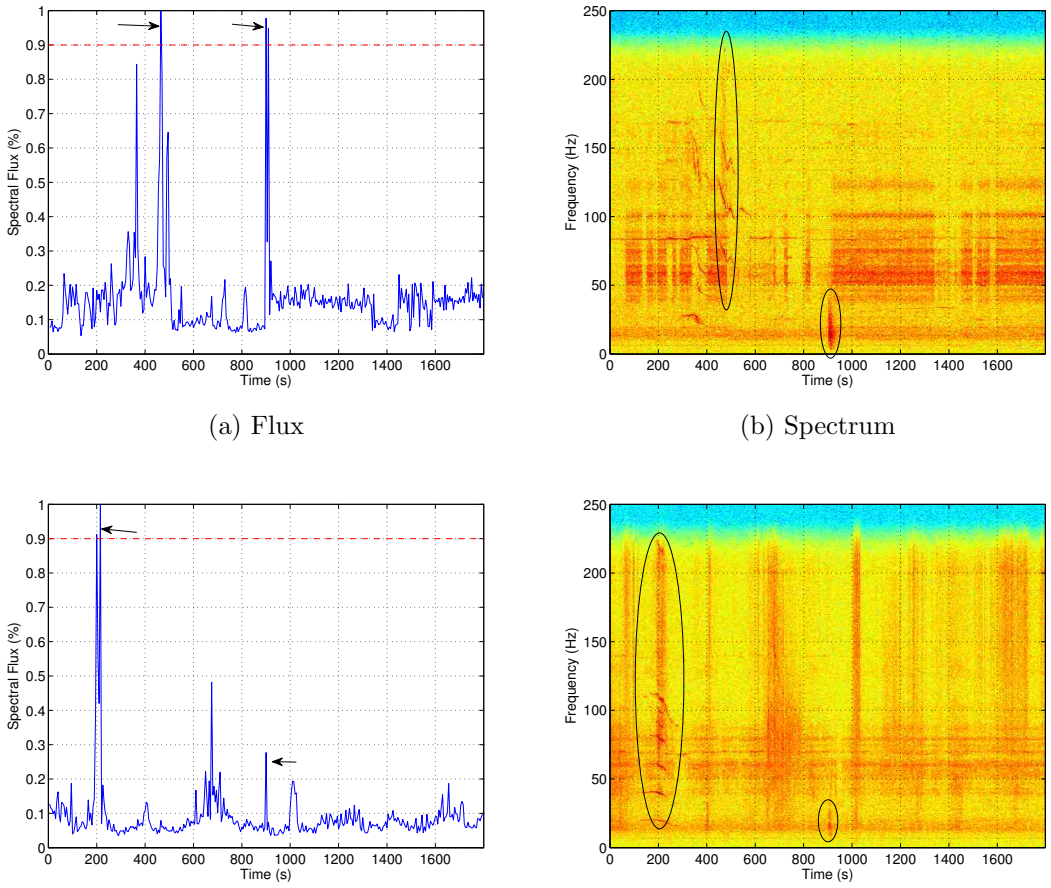


Figure 2.7: Results from testing our voting based classification scheme over the entire season. A perfect classifier is point (0,1). The blue line indicates random classification.

Spectral flux is simply the Euclidean distance between two successive spectral frames. In our workflow, we calculated a 2048-bin FFT for each five second frame and determined the Euclidean distance (spectral flux) between consecutive frames. An event was selected if there was an instantaneous increase of spectral flux above a predetermined threshold.

In our workflow, we used spectral flux to select events as follows. First we divided the data into consecutive five minute subsets. Second, for each five minute subset of data, we calculated the spectral flux between all consecutive five second spectral frames. Third, we normalized the spectral flux by dividing each value by the maximum flux for that five minute subset. Lastly, we selected the five second frames with normalized spectral flux values above a predetermined threshold (Figure 2.8). We experimented with thresholds from 10% to 90% instantaneous increase in spectral flux (Table 2.5). In all threshold cases, one slab avalanche was not selected; this unselected slab avalanche occurred on January 23^{rd}, 2011 and was 5.77 seconds long.



(a) Flux

(b) Spectrum

(c) Flux

(d) Spectrum

Figure 2.8: Spectral flux based event selection was used to identify significant events. (a) Spectral flux was calculated for each five second frame (the red dotted line represents the 90% threshold). (b) An unknown event around time 500 and confirmed slab avalanche at time 900. (c) Spectral flux computed for a different day. (d) An unknown event around time 200 and unconfirmed sluff avalanche at time 900.

For an entire seasons worth of data, using spectral flux to select interesting events resulted in a 97.6% reduction in the problem space: from 1,264,648 to 30,822 five second frames. Our results show that with a spectral flux threshold of 90%, we selected 97% (32 of 33) slab avalanches and 70% (246 of 352) of the smaller sluffs. It is important to note that avalanche forecasters care mostly about automatically identifying the confirmed slab avalanches; in other words, it is ok for our automated workflow to miss some of the smaller sluffs in favor of improved results on detecting slabs.

Table 2.5: Results from experiments testing the spectral flux based event selection. With a strict threshold of 90% instantaneous increase in spectral flux, we select 97% (32 of 33) of slab avalanches and 70% (246 of 352) of smaller sluffs.

Threshold	**Slabs (%)**	**Sluffs (%)**	**Total Events**
0.10	97	96	251,391
0.20	97	93	234,063
0.30	97	87	201,290
0.40	97	83	165,563
0.50	97	81	129,816
0.60	97	77	96,207
0.70	97	73	67,301
0.80	97	72	45,599
0.90	**97**	**70**	**30,822**

After using spectral flux to select events of interest, we trained and tested all 12 classifiers on each of the nine subsets of data (one for each of the nine thresholds). The classification experiments used the same workflow as described previously. The results show that by increasingly limiting our problem space to only the most significant spectral events, i.e., higher thresholds, we increased the precision of all classification models (i.e., compare Table 2.3 vs. Table 2.6). For example, before using spectral flux based event selection, the random forest classifier had 2.3% precision (see Table 2.3); after using spectral flux based thresholding to select energetic events, the random forest's precision increased to 10.2% (see Table 2.6).

To clarify, the results presented in Table 2.6 include *all* five-second frames from the entire season of data. Specifically, in addition to the frames selected using spectral flux (see Table 2.5), our analysis includes the frames not selected as well. This includes over one million true negatives, pertaining to the unselected non-avalanche frames that were ignored using spectral flux based event selection. Additionally, our results include two false negative frames from the unselected slab avalanche that occurred on January 23^{rd} (length 5.77 seconds).

The best performing model was the decision stump classifier, reaching 99.8% accuracy, 88.8% recall, and 13.2% precision when the threshold equals 90% (see Table 2.7 and Figure 2.9 for details). The increased recall and precision makes our model much more reasonable.

Table 2.6: Mean results from testing 100 iterations of 12 machine learning algorithms on a single sensor's data when using spectral flux based event selection with a 90% threshold. Each algorithm was trained on 10% of the selected avalanches and an equal number of non-avalanche events selected using cluster-based subsampling. Testing occurred on all data not used for training and results include *all* frames, both selected and unselected, from the entire season.

Algorithm	Accuracy	Recall	Specificity	Precision
ANN	0.998	0.875	0.998	0.081
Bayes	0.998	0.896	0.998	0.104
BayesNet	0.998	0.892	0.998	0.079
CART	0.998	0.889	0.998	0.123
Fuzzy	0.998	0.897	0.998	0.082
GaussProc	0.998	0.905	0.998	0.099
J48	0.998	0.879	0.998	0.107
KNN	0.995	0.822	0.996	0.036
RandForest	0.998	0.888	0.998	0.102
RIPPER	0.998	0.881	0.998	0.117
STUMP	0.998	0.888	0.998	0.132
SVM	0.998	0.890	0.998	0.105

We reiterate that our analysis includes *all* frames from the entire season of data (and not just the frames selected using spectral flux). Specifically, the results presented in Table 2.7 and Figure 2.9 include not only the selected frames, but also the unselected frames as well; such unselected frames include over one million true negatives from unselected non-avalanche frames and two false negative frames pertaining to the unselected slab avalanche.

We note that performing spectral flux based event selection on all seven sensors and employing voting based classification proved unsuccessful; in other words, there was little consistency of selected events between all sensors. That is, an event selected as an avalanche by sensor one would very rarely be selected by a different sensor. Thus, because few events were shared between all sensors, voting could not be conducted effectively.

2.4.5 Observation: Reducing the Majority Class

Our results indicate that using spectral flux to only select frames with a large instantaneous increase in spectral energy helped improve the precision of each classifier. We hy-

Table 2.7: Mean results for a decision stump classifier trained on events selected using spectral flux based event selection with different thresholds. Results include *all* frames, both selected and unselected, from the entire season.

Decision Stump Classifier				
Threshold	Accuracy	Recall	Specificity	Precision
0.10	0.988	0.850	0.988	0.039
0.20	0.988	0.854	0.988	0.040
0.30	0.992	0.846	0.992	0.051
0.40	0.992	0.864	0.992	0.052
0.50	0.994	0.876	0.994	0.060
0.60	0.995	0.872	0.995	0.072
0.70	0.997	0.873	0.997	0.089
0.80	0.998	0.876	0.998	0.111
0.90	0.998	0.888	0.998	0.132

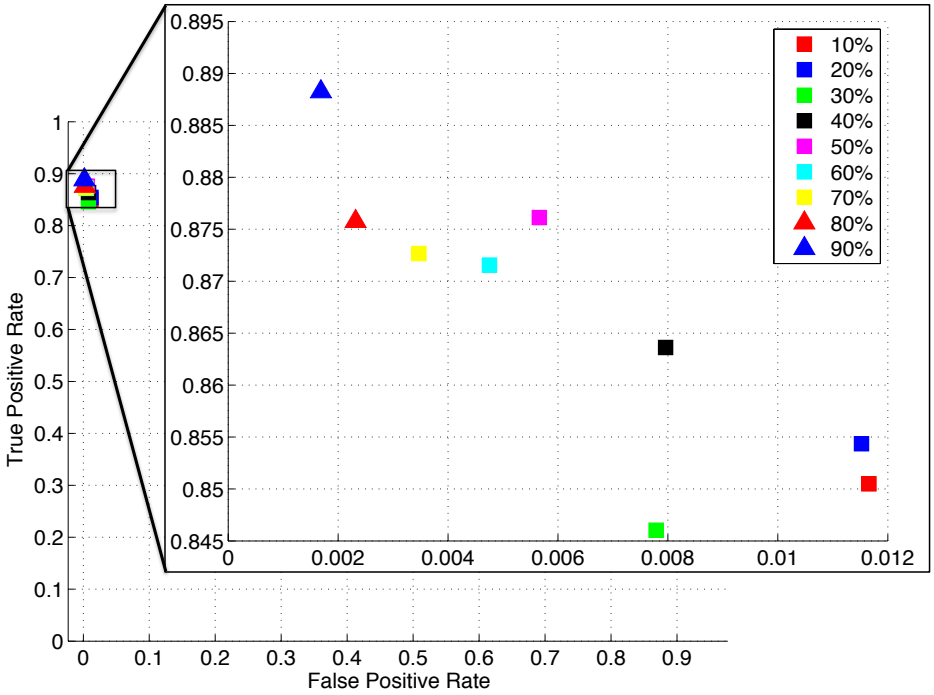


Figure 2.9: Results from training a decision stump classifier using various thresholds for the spectral flux based event selection. Results include *all* frames, both selected and unselected, from the entire season.

pothesize that this improvement in classifier precision is due to the significant reduction in the number of majority (non-avalanche) class objects (see Table 2.5). To test our hypoth-

esis, we trained and tested a classifier 100 times using different proportions of the majority class in the test data sets. In other words, we trained a decision stump classifier using the stratified cluster-based subsampling method described previously and tested it using data sets with subsequently fewer randomly selected majority class objects (e.g., 10%, 20%, ..., 90% of original), ultimately testing with balanced test sets containing the same number of avalanche and non-avalanche events (99.83% reduction). It is important to note that we did not employ spectral flux based event selection in this effort.

The results (see Figure 2.10) show the effects of reducing the majority class in the test dataset, and reveal a strong relationship between reducing the size of the majority class and increasing classifier precision. Our results demonstrate the importance of balancing test datasets before attempting classification, in order to decrease the number of false positives. More specifically, we should utilize techniques to help reduce the majority class before classification.

2.5 Extended Work

Though we obtained successful classification accuracies, the relatively low precision rates of our model indicates that there is still room for improvement. As results show in Section 2.4.5, reducing the number of majority class instances improves classifier precision. Thus, in an attempt to increase classifier precision further, we extend our work to investigate ways to reduce the number of non-avalanche events selected before training and classification begins. In this section we share *preliminary* work regarding our plans for more intelligent event selection and discuss how such event selection could lead to an improved avalanche detection system.

2.5.1 Improving Event Selection

In the workflow presented throughout this chapter, we used spectral flux to select seismic events of interest. Spectral flux was used based on our intuition; avalanches often appeared as sudden energetic increases in the seismic data. Such event selection strategy begs the

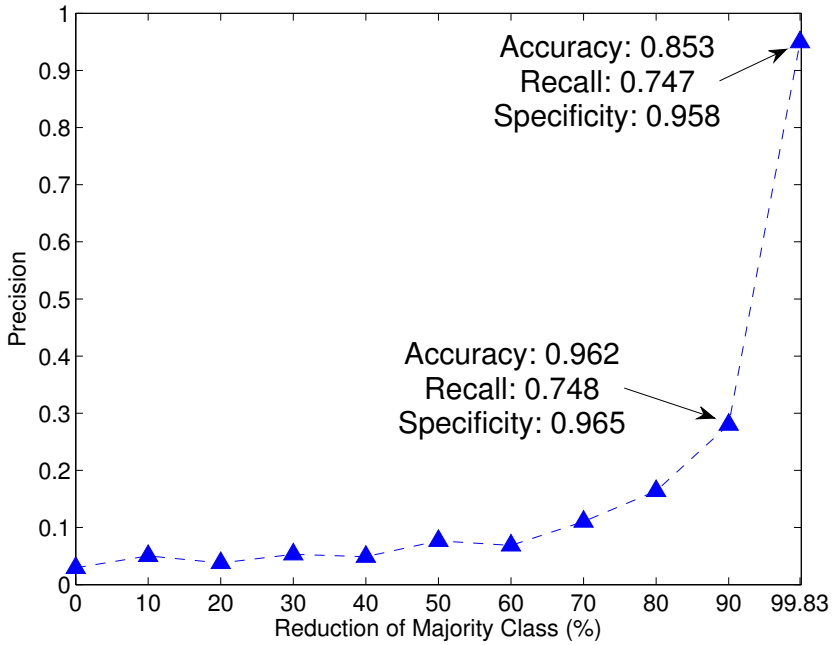


Figure 2.10: Reducing the size of the majority class during testing led to increased precision (and specificity) for our decision stump classifier trained using our pattern recognition workflow; however, we note accuracy and recall results drop dramatically. We did not employ spectral flux based event selection in this work.

questions: "what *other* event selection methods work?" and "what happens if we *combine* such metrics?" In this section, we present preliminary work that demonstrates our plans for future research in automated avalanche detection within seismic data.

To help answer the previous questions, we processed all 385 events (33 slabs and 352 sluffs) in the following manner. First, we extracted five minutes of data containing the known event, with the avalanche occurring at two and a half minutes. Second, we decomposed the five-minute signal into non-overlapping frames, each five seconds in length. Third, for each five-second frame of data, we computed six different metrics common to music signal processing [21]; such metrics are often used to extract events in audio signals. Table 2.8 describes the six metrics used in our preliminary work. For specific information regarding *how* such metrics are calculated, we refer the reader to [20] and [21]. All signal processing was performed using Matlab and the open-source MIRToolbox.

Table 2.8: The six metrics used in our extended analysis of event selection methods. These metrics come from the field of audio signal processing and music information retrieval.

Metric	**Description**
Zerocross	Number of times signal crosses zero.
Flux	Euclidean distance between spectral frames.
Centroid	"Center of gravity" of spectral frames.
RMS	Root-mean-square energy of signal.
Spread	The standard deviation of the spectral frame.
85% Rolloff	The frequency at which 85% of all spectral energy is below.

After computing the six metrics for each five second frame, we sorted the results and selected the frames corresponding to the five highest or lowest values, depending on the metric and our intuition. Specifically, for zerocross, centroid, spread, and rolloff, we selected the five frames responsible for the lowest computed values, as avalanche events generally have significant low frequency energy. For flux and RMS, we selected the five frames with the highest values, as avalanche events typically have large sudden increases in spectral flux and RMS. Figures 2.11 and 2.12 show five minutes of slab and sluff data, respectively, along with the five selected frames for each metric.

The data presented in Figures 2.11 and 2.12 provide clues to improve our automated avalanche detection workflow. For example, the "top" values of four of the six metrics overlap each other 100% in Figure 2.11. Likewise, note that RMS and spread do not overlap the sluff avalanche in Figure 2.12 and only one of the frames selected by flux overlaps the sluff avalanche correctly. Such information will help us differentiate between the slabs and sluffs. For example, a slab avalanche likely has more energy than a sluff, which is indicative of higher RMS values. Also, a slab likely has more concentrated low frequency spectral energy, which would lead to smaller spread, lower rolloff frequency, and lower centroid.

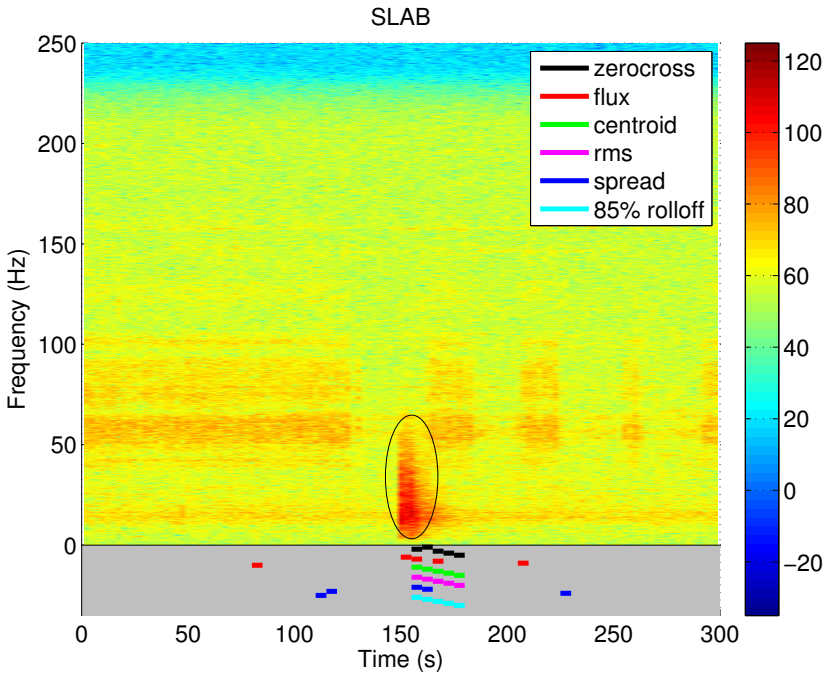


Figure 2.11: An example slab avalanche plotted in the frequency domain, with the five selected frames for each of the six metrics plotted underneath (i.e., below 0 Hz frequency).

2.5.2 Evaluation of Event Selection Methods

We experimented with all combinations of the six metrics, using groupings of size one to six, to identify how well various combinations select slab avalanches within the five minute chunks of data containing slab avalanches (33 total). We focus only on the slab avalanches because, as stated previously, identifying the large, dangerous slab avalanches is of most concern to avalanche forecasting; identifying the sluffs is a bonus. Additionally, focusing on slab avalanches in seismic data is a simplified problem that could help us solve the more difficult task of identifying (and perhaps differentiating) slabs *and* sluffs. Thus, the results presented in this section pertain to the ability to select the slab avalanches only; identifying sluff avalanches is part of our future work.

As mentioned, we experimented with all combinations of zerocross (ZC), flux (FL), centroid (CR), root-mean-square (RMS), spread (SP), and 85% rolloff (RO). We evaluated groupings of size one through six metrics, with all possible combinations in each instance;

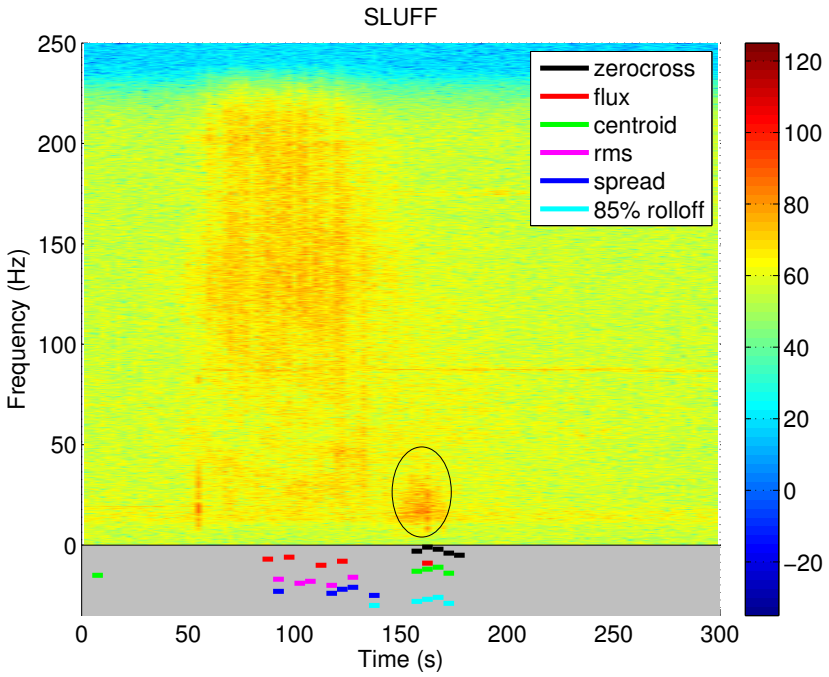


Figure 2.12: An example sluff avalanche plotted in the frequency domain, with the five selected frames for each of the six metrics plotted underneath (i.e., below 0 Hz frequency).

we selected five-second frames as events if and only if there was overlap between the metrics used in the evaluation. For example, notice in Figure 2.11 that all six metrics are overlapping in the five-second frame immediately after time 150 seconds; in the case of using all six overlapping metrics to identify events, only one event will be picked from Figure 2.11. Tables 2.9, 2.10, 2.11, 2.12, 2.13, and 2.14 show the results of our evaluation using combinations of size one through six overlapping metrics, respectively.

In the tables presented, "Metric" refers to the event selection method used, "Events Selected (#)" shows how many events were selected based on the overlapping metrics, "Slab Events (%)" show the percentage of selected events that are a slab avalanche (in time), and "Slabs Missed (%)" shows the percentage of slab avalanches that were not identified using the corresponding overlapping metrics. For clarification, good performance would be indicated by few selected events, a high percentage of events within slabs, and 0% missed slab avalanches. In Tables 2.9 through 2.14, the "best performing" combination is noted

with **bold-face**; such combinations were selected as best performers based on having a low number of missing slabs, a high percentage of selected events containing slabs, and a low number of total events selected. Recall that, as shown in Section 2.4.5, reducing the majority class improves the precision of our workflow; as such, it is important that the event selection method pick as few non-avalanche events as possible.

Table 2.9: Results from selecting events using a single metric. 'ZC' is zerocross, 'FL' is spectral flux, 'CR' is spectral centroid, 'RMS' is root-mean-square, 'SP' is spectral spread, and 'RO' is 85% rolloff. The best performing metric is in bold.

Metric	Events Selected (#)	Slab Events (%)	Slabs Missed (%)
ZC	165	56.36	6.06
FL	165	49.09	0.00
CR	165	55.15	6.06
RMS	**165**	**58.18**	**0.00**
SP	165	40.61	6.06
RO	165	56.97	0.00

The results presented in Table 2.9 show that using only a single metric for selecting events has a high percentage of slabs selected (i.e., small percentage of slabs missed), but does so at the cost of more selected events. Since less than 60% of the events selected were slabs, using only a single metric captures many noise events. In other words, selecting five events per five minutes of data picks many extraneous events that, as the results in Section 2.4.5 show, may lead to poor classifier precision. Recall that our evaluation in this analysis is limited to *only* five-minute chunks of data containing slab avalanches; extending this work to a season's worth of seismic data would likely select many more noise events.

The results in Table 2.10 show that combining two metrics is advantageous to selecting events more precisely. In other words, limiting event selection to only events that contain, for example, periods of high RMS *and* low 85% rolloff values will produce fewer total events that are more likely slab avalanches. This trend continues when selecting events based on three, four, or five overlapping metrics (i.e., Tables 2.11, 2.12, and 2.13, respectively). Additionally, note how the percentage of missed slabs (for the top performing combinations)

Table 2.10: Results from selecting events using two metrics. A five-second frame was selected as an event if and only if the frame was identified as an event by *both* metrics. The best performing combination of two metrics is in bold.

Metric	Events Selected (#)	Slab Events (%)	Slabs Missed (%)
ZC,FL	46	84.78	48.48
ZC,CR	114	71.05	9.09
ZC,RMS	103	76.70	9.09
ZC,SP	69	78.26	12.12
ZC,RO	107	74.77	9.09
FL,CR	52	75.00	48.48
FL,RMS	53	77.36	45.45
FL,SP	44	70.45	54.55
FL,RO	52	78.85	42.42
CR,RMS	110	76.36	6.06
CR,SP	82	73.17	9.09
CR,RO	118	74.58	6.06
RMS,SP	96	66.67	6.06
{RMS,RO}	**116**	**73.28**	**3.03**
SP,RO	98	64.29	6.06

increases slowly between two and five overlapping metrics, with a large jump to over 60% missed slabs when all six metrics are overlapping (see Table 2.14).

It is particularly interesting to note that spectral flux was not included among the top performing sets of overlapping metrics in Tables 2.10 through 2.13; in fact, when spectral flux was included as one of the overlapping metrics, the percentage of missed slabs was much higher than when spectral flux was excluded. These results suggest that our intuition to use spectral flux based event selection in Section 2.4.4 may have been flawed; in other words, there appears to be better ways to select events of interest and help reduce the majority class before classification. On a positive note, these results tell us that our pattern recognition workflow can be further improved.

The results in Tables 2.10 through 2.13 show that combining event selection metrics helps reduce the number of overall selected events and increase the number of events with slab avalanches. Such improvement in event selection comes at a price, however; that is,

Table 2.11: Results from selecting events using three metrics. A five-second frame was selected as an event if and only if the frame was identified as an event by *all three* metrics. The best performing combination of three metrics is in bold.

Metric	Events Selected (#)	Slab Events (%)	Slabs Missed (%)
ZC,FL,CR	40	85.00	51.52
ZC,FL,RMS	37	89.19	51.52
ZC,FL,SP	30	86.67	57.58
ZC,FL,RO	37	89.19	51.52
ZC,CR,RMS	94	80.85	9.09
ZC,CR,SP	64	82.81	12.12
ZC,CR,RO	99	79.80	9.09
ZC,RMS,SP	62	83.87	12.12
ZC,RMS,RO	94	79.79	9.09
ZC,SP,RO	66	80.30	12.12
FL,CR,RMS	41	85.37	51.52
FL,CR,SP	34	82.35	54.55
FL,CR,RO	44	86.36	48.48
FL,RMS,SP	36	77.78	57.58
FL,RMS,RO	40	85.00	51.52
FL,SP,RO	37	78.38	54.55
CR,RMS,SP	75	77.33	9.09
{CR,RMS,RO}	**104**	**78.85**	**6.06**
CR,SP,RO	78	76.92	9.09
RMS,SP,RO	80	75.00	6.06

decreasing the total number of events selected likely means that more slab avalanches are missed entirely. For automated avalanche monitoring, these false negatives are problematic and should be eliminated (or at least minimized); though slab avalanches are quite rare, such events are very important to the avalanche forecasting processing. Assuming that an avalanche did *not* occur when one actually did likely leads to imprecise and inaccurate forecasts; in other words, assuming that an avalanche did not happen may lead forecasters to believe that the snowpack is safer than it actually is.

Our preliminary evaluation provides insight into ways to improve event selection before classification takes place. That is, methods to reduce the majority class of non-avalanche events *before* classification likely improves the precision of our workflow (as shown in Section

Table 2.12: Results from selecting events using four metrics. A five-second frame was selected as an event if and only if the frame was identified as an event by *all four* metrics. The best performing combination of four metrics is in bold.

Metric	Events Selected (#)	Slab Events (%)	Slabs Missed (%)
ZC,FL,CR,RMS	35	88.57	54.55
ZC,FL,CR,SP	29	86.21	57.58
ZC,FL,CR,RO	37	89.19	51.52
ZC,FL,RMS,SP	28	85.71	60.61
ZC,FL,RMS,RO	34	88.24	54.55
ZC,FL,SP,RO	29	86.21	57.58
ZC,CR,RMS,SP	61	83.61	12.12
{ZC,CR,RMS,RO}	**90**	**82.22**	**9.09**
ZC,CR,SP,RO	63	84.13	12.12
ZC,RMS,SP,RO	60	85.00	12.12
FL,CR,RMS,SP	32	81.25	57.58
FL,CR,RMS,RO	40	85.00	51.52
FL,CR,SP,RO	34	82.35	54.55
FL,RMS,SP,RO	32	81.25	57.58
CR,RMS,SP,RO	73	79.45	9.09

Table 2.13: Results from selecting events using five metrics. A five-second frame was selected as an event if and only if the frame was identified as an event by *all five* metrics. The best performing combination of five metrics is in bold.

Metric	Events Selected (#)	Slab Events (%)	Slabs Missed (%)
ZC,FL,CR,RMS,SP	27	85.19	60.61
ZC,FL,CR,RMS,RO	34	88.24	54.55
ZC,FL,CR,SP,RO	29	86.21	57.58
ZC,FL,RMS,SP,RO	27	85.19	60.61
{ZC,CR,RMS,SP,RO}	**60**	**85.00**	**12.12**
FL,CR,RMS,SP,RO	32	81.25	57.58

Table 2.14: Results from selecting events using all six metrics. A five-second frame was selected as an event if and only if the frame was identified as an event by *all six* metrics. The best performing combination of all six metrics is in bold.

Metric	Events Selected (#)	Slab Events (%)	Slabs Missed (%)
ZC,FL,CR,RMS,SP,RO	27	85.19	60.61

2.4.5). This section illustrates that combining event selection metrics works well, because it helps reduce selecting spurious noise events that may, for example, have both high spectral flux and high spectral centroid values. Such events (i.e., high flux and centroid) are usually indicative of high energy, broad spectrum noise events caused by, for example, airplanes or bombs.

2.5.3 Future Work

Our recent results regarding improved event selection suggests much future work. First, we intend to experiment with various permutations of the event selection parameters. In our current work, we selected five events (of length five seconds) within each five minute chunk of data. We plan to explore how modifying such parameters effects the results; specifically, we plan to vary the following parameters: the number of selected events (e.g., 5, 50, 500 events), frame size (e.g., 5, 10, 15 seconds), and data size (e.g., 5, 30, 60 minutes). In addition to varying the event selection parameters, we plan to evaluate our event selection methodology on larger sets of data (e.g., an entire season).

Another aspect of future work that we plan to investigate concerns combining our event selection methodology with other types of information, such as: 1) weather data (e.g., new snow, wind direction and speed), 2) airplane flight paths (e.g., time, date, and direction), and 3) avalanche bombing schedules. Such information could help identify times of "prime" avalanche conditions and eliminate noise events (e.g., airplanes and bombs). Additionally, we plan to utilize data from multiple geophones and sensor geometry to infer acoustic wave velocity and angle of arrival. This information could help us identify the source location and speed of the acoustic energy, leading to more precise classification strategies.

Finally, we intend to investigate semi-supervised and unsupervised avalanche detection methods; that is, we are interested in creating avalanche detection models that do not fully rely on training data picked by an expert human. In a semi-supervised modality, for example, the classification model would periodically ask a human expert for guidance (e.g., is this an avalanche?) and adjust accordingly. Perhaps we can use crowd-sourcing to help pick such

events and eliminate the need for tedious training set identification altogether.

2.6 Conclusions

In this chapter, we present a pattern recognition workflow to automatically detect slab avalanches within passive seismic data. Previous attempts to detect avalanches from passive seismic data either required human experts to develop fuzzy logic rules [13–16], or failed to reach favorable classification accuracies [18]. This work improves upon the current literature by reaching 99% classification accuracy (with over 13% precision) using automated machine learning algorithms to train classification models. Additionally, we illustrate that event selection is a desirable pre-processing step to increase classifier precision in avalanche detection; classification models perform better when there are fewer benign non-events to classify. Finally, we show preliminary work evaluating combinations of different event selection methods; combining event selection metrics will likely improve the precision of our pattern recognition workflow.

CHAPTER 3

ON-MOTE COMPRESSIVE SAMPLING

Wirelessly collecting continuous geophysical data (e.g., seismic and infrasound) at moderately high sampling rates (e.g., 100 to 1000 Hz) requires a more intelligent approach than simply transmitting all the data. As we know, the radio on a wireless mote platform typically consumes orders of magnitude more power than all the other elements (e.g., ADC and CPU) [8]. Thus, to avoid costly power solutions such as large batteries and solar panels, it is imperative to reduce the amount of data that is transmitted wirelessly. In this chapter, we explore on-mote compressive sampling as a means of data reduction for moderate to high sample rate wireless sensors. In particular, we present our on-mote compressive sampling algorithm and compare it experimentally to the two existing on-mote compressive sampling algorithms that are in the literature.

This chapter is organized as follows. First, we summarize the existing literature regarding on-mote compressive sampling. Second, we describe our implementations of the two existing on-mote compressive sampling algorithms and our novel approach. Third, we describe experimental results from compressively sampling a sparse sinusoid using the three on-mote compressive sampling algorithms. Fourth, we analyze the power consumption of these three algorithms. Fifth, we show results from simulating the algorithms on a real-world passive seismic dataset. Lastly, we conclude with a summary of our efforts.

3.1 Background

In this section we describe the current body of knowledge regarding the mechanics of compressive sampling (or sensing) in wireless sensor networks. For a more thorough explanation of compressive sampling, please see Appendix A.

In general, research regarding compressive sampling in wireless sensor networks has focused on spatially sparse signals, where compressive sampling is achieved in a distributed

manner (e.g., [28–31]). For example, the authors of [28] developed a compressive wireless sensing (CWS) technique that attempts to estimate spatially k-sparse signals from a dense distributed wireless sensor network. In CWS, at any iteration of time, only M (randomly selected) of N nodes transmit a sensed sample to a base station. Node selection is based on a random Rademacher measurement matrix that each node and base station constructs independently (assuming perfect synchronization and pseudo-random number generation). The authors of [28] present results from several simulations where the CWS algorithm was used to estimate a spatially sparse 256×256 pixel image with a node representing each pixel. In addition to showing successful signal estimates from very few measurements (i.e., 1600 of ∼65,000 pixels), the authors also summarize the power savings of CWS due to reduced radio transmissions (compared to a "transmit all" approach).

Though the results of [28] show how compressive sampling in wireless sensor networks can reduce power consumption, the CWS algorithm (and other approaches, e.g., [29–31]) focus on recovering spatially sparse signals from a dense distributed wireless sensor network. Our approach aims to use compressive sampling to reduce radio transmissions on a *single* wireless node tasked with continuous geophysical sampling at high sample rates.

We are aware of two on-mote compressive sampling algorithms that have been disseminated: 1) Additive Random Sampling (ARS) [32, 33] and 2) Sparse Binary Sampling (SBS) [34]. In this section we present an overview of these two algorithms, as well as the context in which they were tested. In Section 3.2, we provide more specific details regarding our implementation of these two algorithms on an Arduino Fio wireless mote platform. Furthermore, in Section 3.2.3 we present our novel on-mote compressive sampling algorithm, which we call the Randomized Timing Vector (RTV) algorithm.

The authors of [32, 33] implemented an on-mote compressive sampling algorithm called Additive Random Sampling (ARS) on a Mica mote running TinyOS. This algorithm works by waking up at random times (based on an estimated random Gaussian number), then sensing, storing, and transmitting the sample. After sampling, the mote calculates the next random

sleep interval and goes into a low power state. The authors used ARS to compressively sample, partially recover, and automatically detect a 450 Hz sinusoidal audio signal sampled at 10, 20, 30, and 250 Hz. In addition to recovering the audio signal well enough to detect the 450 Hz tone, the authors showed that ARS can significantly reduce power consumption when compared to full Nyquist sampling at 1024 Hz. Interestingly, their power analysis showed that the radio was the *second* biggest power consumer; generating the next random sleep number using emulated floating point arithmetic drew the most power.

For the second algorithm, SBS, the authors of [34] implemented a different on-mote compressive sampling algorithm, called Sparse Binary Sampling, to efficiently collect ECG data on the Shimmer mote platform. In particular, the authors compared their non-adaptive on-mote compressive sampling algorithm to an on-mote implementation of the discrete wavelet transform (DWT), the industry standard for adaptively compressing ECG data. Though the authors reported "inferior" recovered signals from compressive sampling (compared to DWT), they noted reduced radio usage and increased battery longevity with compressive sampling. Briefly, the SBS algorithm works by first sampling all $N = 512$ ECG measurements (at 254 Hz) and storing the data in memory. Next, as is typical of compressive sampling (and described further in Appendix A), a linear transformation is performed by calculating the product $y = \Phi x$, where Φ is a sub-Gaussian measurement matrix with $D = 12$ non-zeros per column, such that each nonzero entry equals $\frac{1}{\sqrt{12}}$. The compressed signal y is transmitted and the full ECG signal is recovered offline using the Matlab based SPGL1 solver (an implementation of ℓ_1-norm minimization).

Although both the ARS and SBS compressive sampling algorithms resulted in suitably recovered signals and increased battery life, there are several inefficiencies that are addressed by our lightweight Randomized Timing Vector (RTV) algorithm. First, ARS necessitates estimating a random Gaussian variable on each iteration, which requires several costly floating point calculations on very limited resources. As we show in Section 3.3, the overhead associated with estimating each Gaussian random variable is substantial, and leads to inef-

ficiencies that inhibit moderately high sampling rates (i.e., 500 Hz or above). In a similar vein, the SBS algorithm requires calculating $y = \Phi x$, which consumes a substantial amount of CPU time and memory on limited mote resources (e.g., 8-16 MHz processor and 2-10 KB of SRAM).

3.2 On-Mote Compressive Sampling Algorithms

In this section, we define the two existing on-mote compressive sampling algorithms in more detail and introduce our compressive sampling algorithm that we call Randomized Timing Vector (RTV). In addition to describing the details of each algorithm, we also provide pseudocode of each algorithm's implementation on the Arduino Fio wireless mote platform. The Arduino Fio wireless mote was selected because, although it has limited resources, it is inexpensive, which makes it ideal for low-cost wireless data acquisition in domains such as ours (i.e., geophysical data acquisition for geohazard monitoring); low-cost wireless data acquisition is appealing to geoscientists because it increases spatial distribution of sensor placements without requiring complex or costly *wired* system installations. Furthermore, we implemented a low-cost "shield" that transforms the off-the-shelf Arduino Fio mote to a research-grade, wireless geophysical sensing tool (see Chapter 5 for details).

3.2.1 Additive Random Sampling

The ARS algorithm [32, 33] achieves compressive sampling by periodically waking up, sensing, storing, and transmitting the data. The sleep periods are determined by a random Gaussian variable $\mathcal{N}(\mu, (r\mu)^2)$, where $r = 0.25$ (per [32, 33]), μ is the average sampling interval ($\mu = \frac{N}{M}F_s$), N is the length of the original signal, M is the length of the compressed signal, and F_s is the sampling interval (e.g., 10 ms for 100 Hz). In the ARS algorithm, the Gaussian random variable $\mathcal{N}(\mu, (r\mu)^2)$ is estimated using a 12^{th}-order Irwin-Hill Gaussian approximate, which sums 12 uniformly distributed numbers then subtracts the value six from the summation, producing a Gaussian number. Using a known seed for the Irwin-Hill approximation, the random sleep time sequence (and subsequent random binary Φ) can be

reconstructed offline during full signal recovery.

Although the ARS algorithm is intuitive, implementing it on real hardware required addressing several subtleties not mentioned in [32, 33]. First, the vast majority of the random Gaussian sleep durations were not aligned to the sampling grid. Specifically, when sampling at 100 Hz, or one sample per 10 milliseconds, a transformed random Gaussian number and sleep duration of, for example, 64.3 milliseconds, would not make sense. Thus, for ARS to work properly, we had to align each estimated Gaussian number to the sampling grid using rounding and modular arithmetic, such that the 64.3 ms sleep time in our example would be aligned to 60 ms. Though crucial for avoiding costly sampling jitter, these alignments required additional calculations and, thus, CPU time.

The second subtlety not addressed in [32, 33] concerned tracking the time elapsed during the next sleep time calculation (i.e., the Irwin Hill Gaussian approximate and alignment to the sampling grid). Failing to incorporate the time required to calculate the next sleep time (i.e., 4.5 ms on average) introduced costly sampling jitter errors. In other words, the 60 ms timer countdown (calculated previously) would not start until 4.5 ms later than intended. Thus, we had to subtract the calculation time from each aligned random Gaussian sleep period. In Section 3.3, we show that this nontrivial calculation time is detrimental to the ARS algorithm. Figure 3.1 provides the pseudocode of our implementation.

3.2.2 Sparse Binary Sampling

To reiterate, SBS [34] works by first sampling and storing N measurements into x, immediately calculating $y = \Phi x$ on the mote platform, and then transmitting the compressed sampling vector y of length M. The sub-Gaussian measurement matrix, Φ, of size $M \times N$, consists of D non-zero entires per column, with the non-zero entries set to $1/\sqrt{D}$. To be consistent with the author's work, we selected $D = 12$ for our experiments [34].

There are at least four different ways we could implement SBS on a wireless mote platform. First, one can simply compute and store Φ in its entirety in RAM; however, storing Φ wastes space (since most elements are zero). Furthermore, mote platforms have limited

```
1. set M, N, r = 0.25, Fs = sampling_interval, seed
2. calculate μ = (N/M)Fs, (rμ)^2
3. initialize y[M], m = 0
4. transmit seed (for recovery)
5. calculate sleepTime = calcSleepTime()
6. initialize timer interrupt for sleepTime

for each timer interrupt:
  sense
  store in y[m]
  m++

  if m == M
    transmit y
    m = 0

  begin stopwatch
  sleepTime = calcSleepTime()
  end stopwatch
  sleepTime -= stopwatch
  reset timer interrupt for sleepTime

calcSleepTime():
  calculate nextSleepTime as 12^th order Irwin Hill
  transform nextSleepTime to N(μ,(rμ)^2) distribution
  align nextSleepTime to sampling grid
  return nextSleepTime
```

Figure 3.1: Pseudocode for the Additive Random Sampling on-mote compressive sampling algorithm.

RAM (e.g., 2-10 KB), which places hard upper limits on the size of Φ. Second, one could implement SBS by calculating each column of Φ on demand. In our experience, this method is too time-consuming to be practical. In particular, calculating $y = \Phi x$ took between 1.7 and 15.8 seconds, depending on the compressive sampling ratio (10% and 90%, respectively), when N=250. This calculation time is problematic at moderate sampling rates (e.g., 100 Hz) and leads to lost data points, even if clever buffering exists.

The third and fourth ways we could implement SBS are similar, and involve storing a sparse representation of Φ in memory. Specifically, one can store the indices of the non-zero entries of Φ as the integer $mN + n$, where N is the number of columns in Φ, and m, n represent the row and column of the nonzero entry $\Phi_{m,n}$. In the third method, this vector of

indices, which we note as Φ_s (length $N \times D$), is stored in RAM; in the fourth method, Φ_s is stored in Flash memory. When Φ_s is stored in RAM, it is calculated during execution time, which allows users to specify how often Φ changes; however, when Φ is stored in RAM, it limits the size of Φ_s, as well as the size of x and y.

As noted, the fourth version of SBS stores Φ_s in Flash memory, which requires hard-coding Φ_s during compile time and implies that Φ_s cannot be modified during execution. Furthermore, storing Φ_s in Flash consumes a lot of space and limits the amount of additional software that can be installed on the wireless mote (e.g., libraries for an SD socket, GPS chip, or external ADC). For example, if N=500, M=450, and D=12, then Φ_s requires 24,000 bytes of Flash memory (i.e., $N \times D$ or 6,000 4-byte numbers holding indices between 0 to 224,999, or $\Phi_{0,0}$ to $\Phi_{M-1,N-1}$). Though storing Φ_s in Flash memory has its drawbacks, we used this method in our experimentation because it allows us to set N=500, D=12, and M up to 450. Figure 3.2 shows the pseudocode of our implementation of SBS.

The SBS algorithm has three main problems. First, calculating $y = \Phi x$ on a wireless mote platform requires a substantial amount of time. Second, SBS requires a considerable memory footprint; that is, at a minimum, both Φ_s and x must be stored in memory. Furthermore, if double buffering is used to store additional samples during radio transmissions, then the vector y must be stored as well. Double buffering refers to filling one buffer (e.g., 250 samples) while the other buffer is being transmitted, and vice versa. We implement SBS with double buffering by 1) sensing and storing $B = \frac{N}{2}$ values in buffer x_1, 2) calculating "half" of $y = \Phi\, x$ (i.e., $y_{1...M} = \Phi_{1...M,1...B}\, x_1$) while sensing and storing the next B values in buffer x_2, 3) calculating the second "half" of $y = \Phi\, x$ (i.e., $y_{1...M} = y_{1...M} + \Phi_{1...M,B+1...N}\, x_2$) while sensing and storing new values in the first buffer x_1, and 4) transmitting y once $y = \Phi\, x$ is fully calculated.

In our implementation, we also saved memory by truncating the 10-bit ADC values to 8 bits and storing the vector y as two byte integers instead of four byte doubles. We note that truncating x and y did not significantly affect simulation results; that is, we were still able to

```
1. generate Φ_s offline; hardcode into Flash memory
2. initialize timer interrupts to sampling interval
3. set counter = 0, D = 12, M, N
4. initialize x[N], y[M]

for each interrupt:
  sense
  store in x[counter]
  counter++

  if (counter == N):
    counter = 0

    calculate y = Φ_s x:
      for m=0:M-1
        y[m] = 0
        for n=0:N-1
          if Φ_s[counter] == m*N+n:
            counter++
            y[m] += x[n] * 1 / √D

      transmit y
      counter = 0
```

Figure 3.2: Pseudocode for the Sparse Binary Sampling on-mote compressive sampling algorithm.

reliably recover the content of the original signals. Third, hardcoding Φ_s in Flash memory severely limits the functionality of the wireless platform. For example, the Arduino Fio has 32 KB of Flash, of which 2 KB are reserved for the boot loader. Of the 30 KB available for storage, 24,000 bytes are required to store Φ_s, leaving about 6 KB for user specific software. With such limited space, users could not, for example, include software for an external SD card, which requires 11 KB of Flash.

3.2.3 Randomized Timing Vector

To address the inefficiencies of both ARS and SBS, we introduce a novel and lightweight on-mote compressive sampling algorithm that we call the Randomized Timing Vector (RTV) algorithm. Our approach does not falter at moderate to high sample rates (tested up to 1000 Hz), does not require any costly floating point calculations, and does not necessitate a

substantial memory footprint.

Our RTV algorithm is based on the following observation: given an $M \times N$ random binary measurement matrix, Φ_b, containing exactly one '1' per row and at most one '1' per column, we can approximate $y = \Phi_b\, x$ without doing a matrix multiplication. In other words, the matrix multiplication $y = \Phi_b\, x$ is equivalent to randomly selecting M samples from the original signal x; row n of x is selected if column n of Φ_b contains a '1'. Furthermore, a random binary measurement matrix Φ_b (size $M \times N$) can be represented by a vector, V_t, of length N. If a column in Φ_b contains a '1' (in any row), then the associated column in V_t also contains a '1' (all other entries will be zero). In other words, the timing vector V_t is a "flattened" version of Φ_b, e.g.,

$$\Phi_b = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \Longrightarrow V_t = \{0, 1, 0, 1, 1\}.$$

This observation has significant implications for on-mote compressive sampling. Specifically, instead of computing $y = \Phi_b\, x$, we simply sample and store y directly. This approach saves both computational time and memory space, i.e., we do not have to perform a costly matrix multiplication nor do we have to store Φ and x. In addition, we only need to sample M elements instead of N.

The RTV algorithm works as follows. First, the randomized timing vector V_t (length N) is initialized using a uniform random number generator with a known seed. In particular, the first M values of V_t are set to '1', the remaining values are set to '0', and V_t is randomly shuffled using the in-place Fisher-Yates algorithm [35]. Next, a timer is initialized for the desired sampling interval (e.g., 2 ms for 500 Hz sampling rate) and a counter is set to zero. For each interrupt, the timing vector is checked (indexed at the counter) and, if a '1' is present, a sample is taken from the ADC and stored in y (of length M). This process continues until the counter reaches N, at which point vector y is transmitted and the counter is reset to zero. Additionally, timing vector V_t is periodically recalculated using a new seed (which is transmitted for offline generation of Φ_b). Figure 3.3 provides pseudocode for RTV.

```
1. set counter = 0, m = 0, N, M
2. initialize timer interrupts to sampling interval
3. initialize y[M], V_t[N], seed
4. generateTimingVector(seed)

for each timer interrupt:
  if V_t[counter] != 0:
    sense
    store in y[m]
    m++
  counter++

  if counter == N
    transmit y
    set counter = 0, m = 0
    set seed = newSeed (periodically)
    generateTimingVector(seed) (periodically)

generateTimingVector(seed):
  transmit seed (for recovery)
  seed random number generator
  for i=0:M-1
    V_t[i] = 1

  for i=M:N-1
    V_t[i] = 0

  //Randomly Shuffle using Fisher-Yates
  for i=N-1:1
    j ← random number between 0 and i
    if i != j
      swap(V_t[i], V_t[j])
```

Figure 3.3: Pseudocode for our novel and lightweight Randomized Timing Vector on-mote compressive sampling algorithm.

3.3 Experimentation and Results

In this section, we describe our experimentation to compare three on-mote compressive sampling schemes: 1) ARS, 2) SBS, and 3) RTV, our proposed new algorithm. This section is organized as follows. First, we explain how we tested the algorithms using a sparse sinusoid produced by a signal generator. Second, we describe the power consumption of the three algorithms, and show the tangible savings that on-mote compressive sampling offers. Lastly, to test the viability of the algorithms in a real-world setting, we evaluate the three algorithms

on geophone (i.e., passive seismic) data containing traces of avalanches events.

3.3.1 Signal Recovery

We tested the three on-mote compressive sampling algorithms using a signal generator and Arduino Fio's 10-bit analog to digital converter (part of the ATMEL ATmega328P microcontroller). The signal generator was used to produce three different 1.65V peak to peak sine waves with a 2V bias. Such input signals are clearly sparse in the frequency domain and provide simplistic inputs with which to test and validate on-mote compressive sampling algorithms (e.g., Figure 3.4).

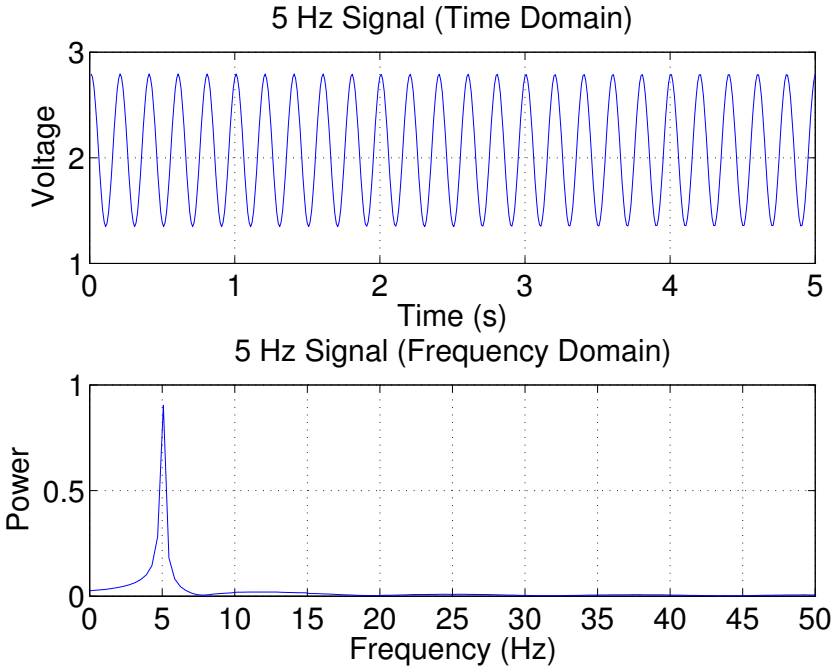


Figure 3.4: We tested the three on-mote compressive sampling algorithms using sinusoids created by a signal generator. Each sinusoid tested (i.e., 5 Hz, 50 Hz, and 351 Hz) was sparse in the frequency domain.

In our experiments, we tested three different sets of inputs and mote sampling rates, i.e., a 5 Hz sine wave sampled at 100 Hz, a 50 Hz sine wave sampled at 500 Hz, and a 351 Hz sine wave sampled at 1000 Hz. For each set, we experimented with 10 different seeds to calculate Φ both on-mote and offline. Moreover, compressive sampling was only performed

on the first N=500 samples. Thus, we were able to ignore the data loss problems associated with the SBS algorithm (see Section 3.2.2 and Section 3.3.2 for details).

Baseline signals were collected using the same mote hardware and full (100%) sampling for each of the three sets of signals and sampling rates. All compressively sensed signals were recovered offline using ℓ_1-norm minimization assuming sparsity in the frequency domain. Recoveries were performed in Matlab using the open-source ℓ_1-magic toolbox [36].

We calculated the normalized root-mean-square error (NRMSE) between the recovered and the original (baseline) signals. NRMSE, which represents the root-mean-square error as a percentage of the original signal's range, is defined as

$$NRMSE = \frac{\sqrt{mean((x - \widehat{x})^2)}}{max(x) - min(x)},$$

where x is the original signal and $\widehat{x}$ is the estimated recovery. To account for phase differences between the original and recovered sinusoid signals, the NRMSEs were calculated in the frequency domain. Specifically, we calculated a 256-bin Fast Fourier Transform (FFT) and estimated the NRMSE based on the frequency spectrums. Figures 3.5, 3.6, and 3.7 plot the mean NRMSEs and 95% confidence intervals of each algorithm and compressive sampling rate. The NRMSE results for the ARS algorithm in Figures 3.6 and 3.7 are incomplete due to high rates (i.e., $> 90\%$) of infeasible sleep times. We evaluate infeasible sleep times further in the following discussion.

There are several trends worth noting. First, the ARS algorithm stopped functioning at high sampling rates and compressive sampling percentages. This trend occurred due to infeasible sleep times, i.e., when the time required to calculate the next random Gaussian sleep interval (about 4.5 ms) was greater than the actual sleep interval calculated. To illustrate this trend, Figure 3.8 shows the mean percentage of total samples that were 'OK', meaning the calculated sleep interval was greater than the time required to calculate it.

The second trend is that, when sampling at 500 Hz and 1000 Hz, the ARS algorithm performed the worst, with the highest NRMSEs in all cases. These results are most likely

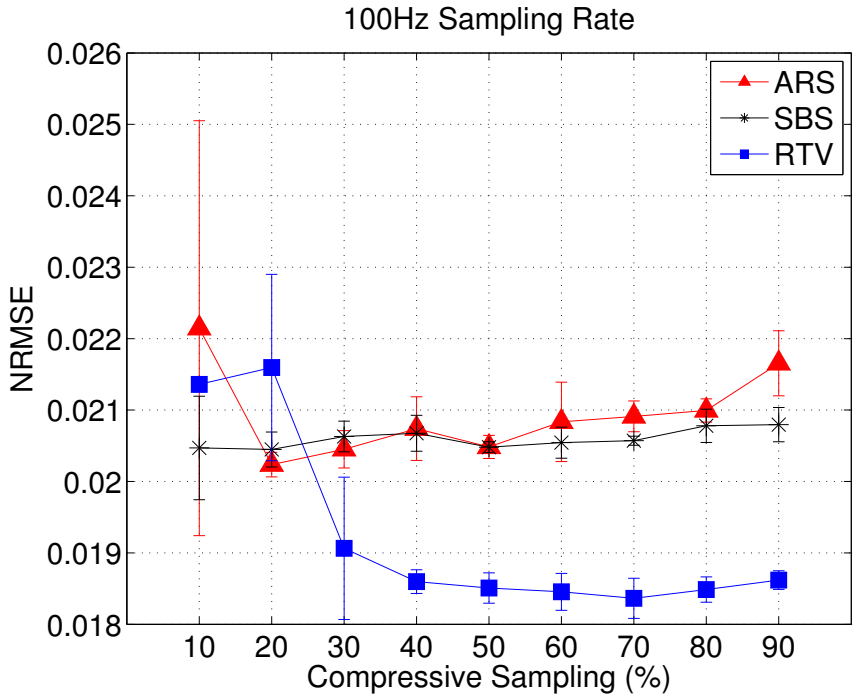


Figure 3.5: Mean NRMSE results (with 95% confidence intervals) of three on-mote compressive sampling algorithms tested with a 5 Hz sine wave sampled at 100 Hz.

due to sampling jitter; sampling jitter is the error introduced by ARS each time the timer interrupts are stopped, recalculated, and restarted. These errors exist despite our efforts to estimate and offset this error (with microsecond precision).

The last trend found in the NRMSE results is that our RTV algorithm performed best overall and had the lowest NRMSEs when the compressive sampling rate is greater than 20%. We expected the SBS algorithm would perform the best, given that the Φ matrix in SBS contains D=12 non-zero entries per column, instead of one '1' per row (as in RTV). In other words, in SBS, each value in y contains more information regarding the entire signal x, since more of x is projected onto y during multiplication. We hypothesize that SBS with $D = 12$ performs worse than RTV because the sine wave signal is 'too' simple, and SBS based ℓ_1-norm recovery tries to resolve a lot more information than is necessary. We confirmed this hypothesis empirically; as signal x increases in complexity (i.e., sinusoids with random frequencies added together), SBS performs slightly better than RTV. For example,

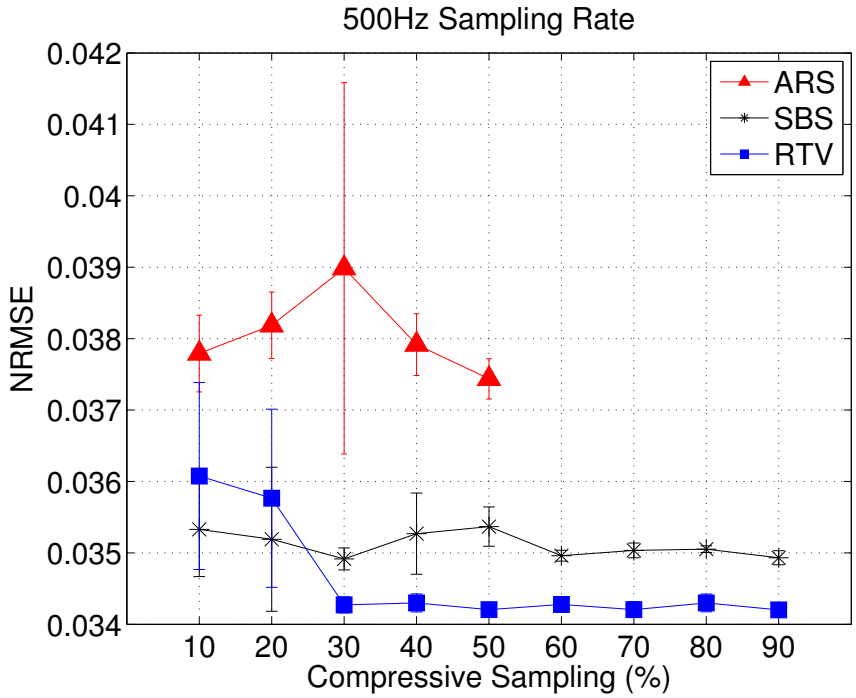


Figure 3.6: Mean NRMSE results (with 95% confidence intervals) of three on-mote compressive sampling algorithms tested with a 50 Hz sine wave sampled at 500 Hz.

SBS recovers a simulated 12-sparse sinusoid signal (i.e., 12 distinct sinusoids added together) slightly better than RTV. These signals, however, are simulated signals. In Section 3.3.3, we compare the algorithms on real-world data.

3.3.2 Power Analysis

Next, we analyzed the three compressive sampling algorithms in terms of power consumption. In particular, we measured voltage differences across a 10.1Ω resistor in series between the wireless mote and ground. Using Ohm's law, we calculate the average current draw as the measured voltage divided by the resistance. For our experiments, we used a National Instruments USB-6218 DAQ and LabView to record the voltages at 10,000 Hz for 60 seconds per test. Specifically, we evaluated power consumption of an Arduino Fio wireless mote equipped with an XBee Pro 802.15.4 long range radio (up to one mile line of sight). Such a powerful radio was chosen to mimic real world wireless geophysical sensing

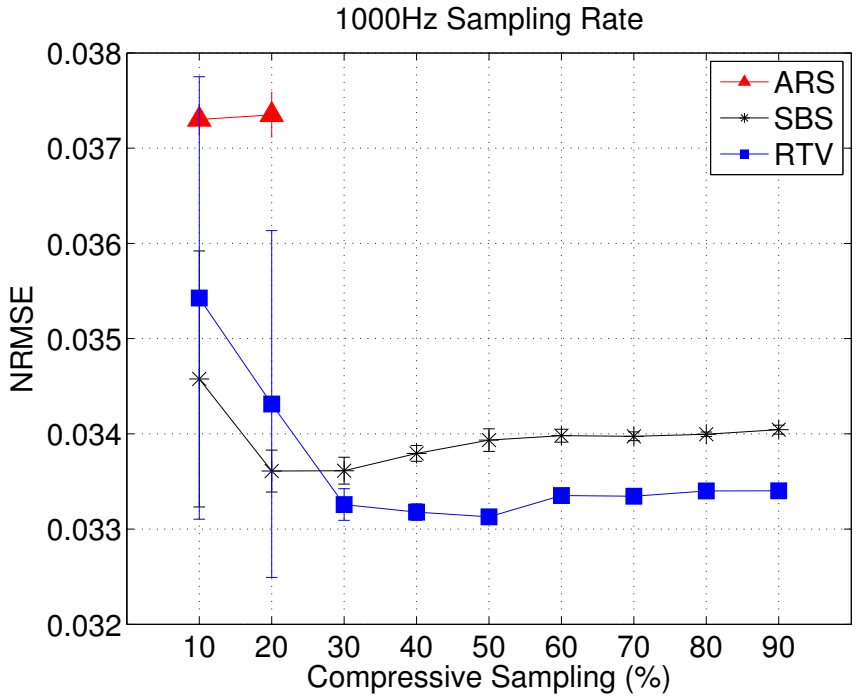


Figure 3.7: Mean NRMSE results (with 95% confidence intervals) of three on-mote compressive sampling algorithms tested with a 351 Hz sine wave sampled at 1000 Hz.

applications that require a sparse spatial distribution of sensors.

We collected voltages and estimated the average current draw of each algorithm with 10% to 90% compressive sampling at 100 Hz sampling rate. For comparison, we also collected data of a wireless mote running full sampling. Though not detailed here, we note that the algorithms were tweaked to reduce power by sleeping the radio and using a double buffering scheme to maximize payloads. Sleeping the radio is evident in Figure 3.9, which shows the current draw for full sampling over a 50 second period (after initialization). Furthermore, double buffering can be seen in Figure 3.9 as well, i.e., there are four transmissions every ten seconds, which equates to 1000 samples at 100 Hz (four 250 sample buffers per 10,000 milliseconds).

For comparison, Figure 3.10 shows the current draw of the RTV algorithm at 20% compressive sampling. The power savings are quite clear; RTV transmits less frequently and, thus, requires less power. In other words, the transmissions are farther apart in RTV because

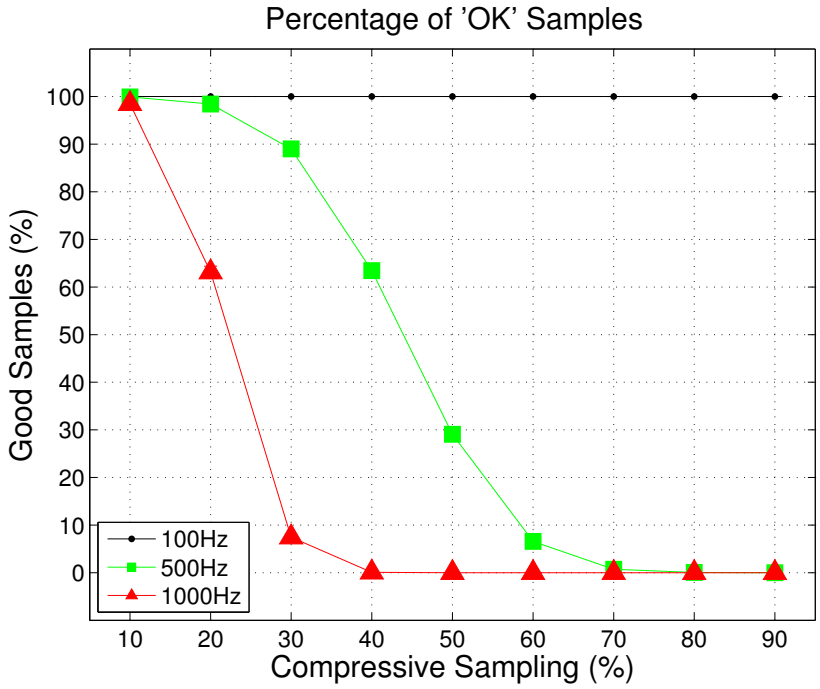


Figure 3.8: Mean percentage with 95% confidence intervals of valid calculated random sleep time intervals for the ARS algorithm. A sleep time was marked valid if the calculated interval was greater than the time required to calculate it. We note the 95% confidence intervals are always smaller than the bullet that represents the mean.

it takes longer to fill a 250 sample buffer when storing only a fraction of the total signal.

We also analyzed the current draw to determine how much power is saved by performing on-mote compressive sampling. Figure 3.11 shows the expected lifetime of a 6.6 Ah battery used to power a wireless mote running the various algorithms. The results show that, when compared to full sampling (FS), on-mote compressive sampling will increase the longevity of the wireless mote by significantly extending the battery life. For example, a mote running the RTV algorithm with 40% compressive sampling will last three times longer than a mote using full sampling.

The primary reason for the prolonged longevity of the mote is due to decreased radio usage. The RTV and ARS algorithms save power by transmitting less frequently; it takes more time to fill a 250 sample buffer (which equals five compressed signals y when $M = 50$) when only storing, for example, 40% of the samples. Though the SBS algorithm transmits

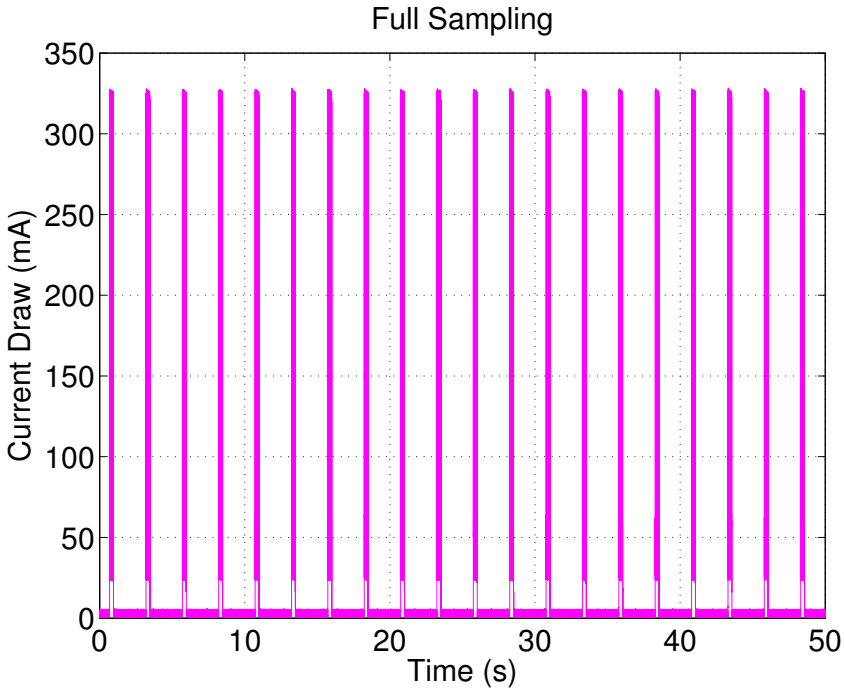


Figure 3.9: The current draw of an Arduino Fio wireless mote performing full sampling and double buffering at 100 Hz.

less data than full sampling (i.e., $M \ll N$), the SBS algorithm does not transmit less frequently than full sampling; that is, SBS must compute $y = \Phi\, x$ and transmit y whenever x is full. Lastly, RTV and ARS further save power by not sampling all N elements of signal x, which eliminates several analog to digital conversions.

It is not surprising that, of the three on-mote compressive sampling algorithms, SBS performed the worst in terms of power savings. This is due to three reasons. First, SBS requires transmitting y each time $y = \Phi\, x$ is calculated, which occurs after N samples are stored in x. These transmissions increase power consumption because, compared to ARS and RTV, the radio spends more time transmitting and less time in a low power state. Second, SBS necessitates calculating $y = \Phi\, x$, which also decreases the amount of time the microcontroller can spend in a lower power 'idle' state. Lastly, SBS requires storing all N samples of signal x, which requires N analog to digital conversions per iteration; ARS and RTV, on the other hand, only convert and store M samples.

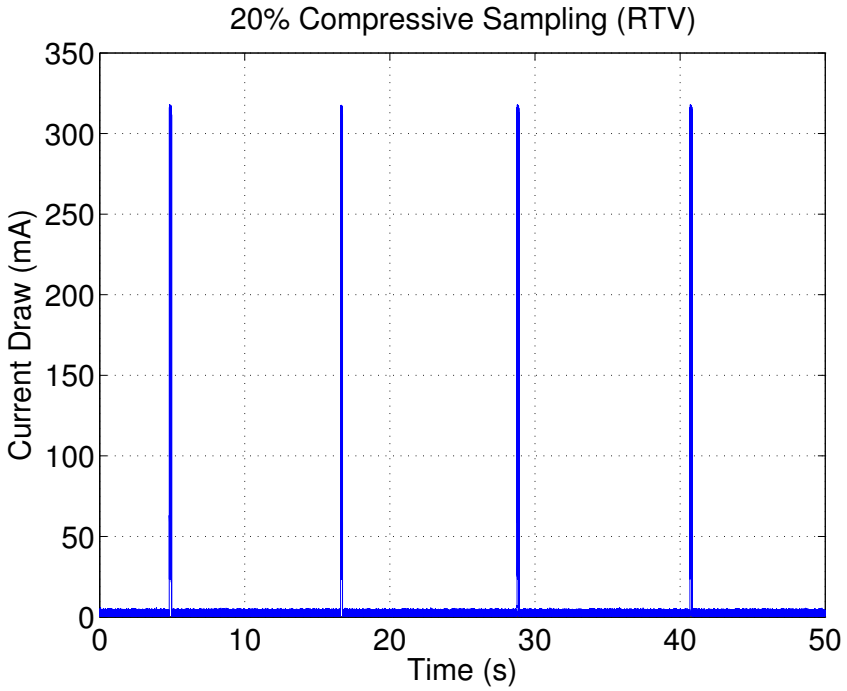


Figure 3.10: The current draw of an Arduino Fio wireless mote running the RTV algorithm with 20% compressive sampling and double buffering at 100 Hz.

Additionally, although we implemented double buffering and calculated subsections of $y = \Phi\, x$ independently, the SBS algorithm loses samples of the original signal (especially at moderate to high sampling rates). This loss happens for two reasons. First, due to the nature of matrix multiplication, we cannot transmit y until the entire $y = \Phi\, x$ is calculated; $y = \Phi\, x$ must be computed before new samples can be stored in the second half of x. Second, the time required to calculate the second half of $y = \Phi\, x$ and transmit y is significant, i.e., between 349.8 and 2,166.5 ms (on average), depending on the compressive sampling percentage. This time delay is problematic because the secondary buffer will fill before the calculation and transmission of y are complete. For example, when sampling at 1000 Hz, the 250 sample buffer will fill in 250 ms, which is less than the time required to compute and transmit y with 10% compressive sampling (i.e., 349.8 ms).

Figures 3.9 - 3.11 illustrate that, compared to ARS, SBS, and full sampling, the RTV algorithm saves the most power. RTV reduces power consumption by transmitting less

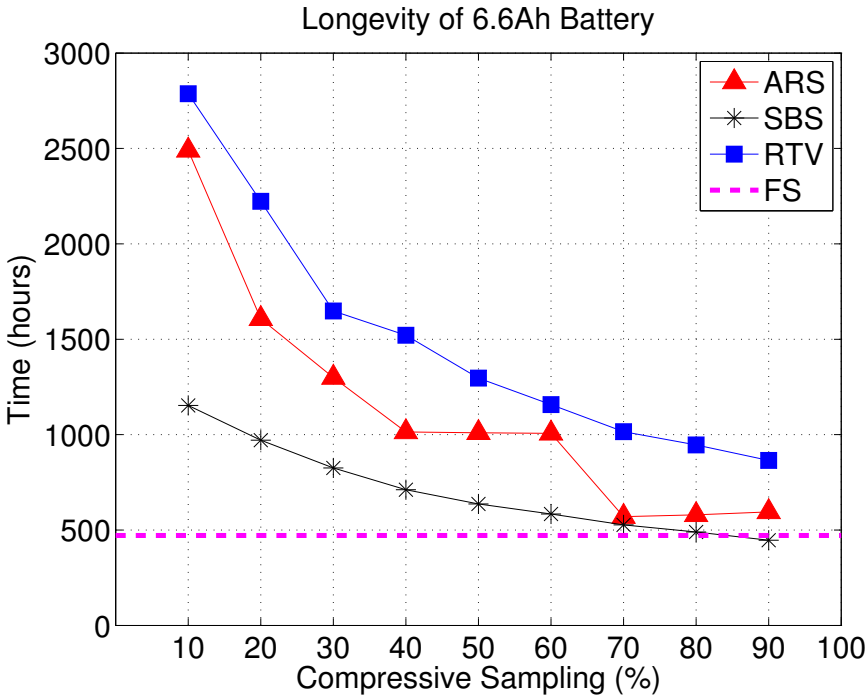


Figure 3.11: The estimated longevity of a 6.6 Ah battery used to power a wireless mote running compressive and full sampling (FS).

frequently, eliminating costly floating point calculations, and reducing the number of required analog to digital conversions. We next evaluate how the three on-mote compressive sampling algorithms fare when simulated on a real-world data set.

3.3.3 Evaluation on Real Data

Our last set of experiments evaluated the three on-mote compressive sampling algorithms on real world passive seismic data. The data was collected using a *wired* geophone array to record avalanches above Davos, Switzerland during the 2009-2010 winter season. More information regarding the deployment and subsequent geophysics research on the data can be found in Chapter 2 and [9]. We selected this real-world dataset to test whether the three compressive sampling algorithms could be used to reduce power consumption without losing the ability to recognize large avalanches in the geophone data. More information regarding our automated avalanche detection workflow can be found in Chapter 2.

Though using ARS and SBS at 500 Hz sampling rate is somewhat infeasible,[4] we tested all three algorithms (ARS, SBS, and RTV) for the sake of comparison. We note that the RTV algorithm functions properly at this moderately high sampling rate. In addition, we did not simulate the signal loss that occurs with the SBS and ARS algorithms. In this regard, these simulations represent the best case scenarios of on-mote compressive sampling; that is, we assume we have suitable memory for double buffering and enough processor speed to calculate either the next sleep time (for ARS) or $y = \Phi\, x$ (for SBS) quickly.

As discussed in Chapter 2, the data set contains 33 large avalanche events (called slabs) and 352 small avalanche events (called sluffs) recorded in the mountains of Switzerland. For our simulation and analysis, we focused on compressively sampling and recovering only the 33 slab avalanches. Specifically, we processed and analyzed 33 unique five-minute frames, where the avalanche started two and a half minutes into the frame (e.g., top of Figure 3.12). We note that many of the slab avalanches are sparse in the frequency domain (e.g., bottom of Figure 3.12), which gives us a compelling reason to use compressive sampling as a means of data reduction. Though not shown here, some avalanches were quite noisy, with significant energy across the entire frequency spectrum.

As mentioned, we simulated the three on-mote compressive sampling algorithms on each of the 33 5-minute frames of passive seismic data containing large avalanche events. In particular, we simulated compressive sampling on each 5-minute signal by compressing and recovering the data one second at a time (i.e., N=500). To recover the signal, we used ℓ_1-norm minimization assuming sparsity in the frequency domain.

For each avalanche signal, we ran the three algorithms with nine different compressive sampling rates, from 10% to 90% (with 10 different seeds used to generate Φ). Thus, each simulation trial of an on-mote compressive sampling algorithm resulted in 2970 estimated signals (i.e., 33 slabs, 9 percentages, 10 seeds). For the SBS algorithm, we generated Φ only

[4] ARS is infeasible at moderately high sampling rates because the calculation of the next random Gaussian interval takes longer than the sleep interval calculated. SBS is infeasible because the time to calculate and transmit $y = \Phi\, x$ results in lost data.

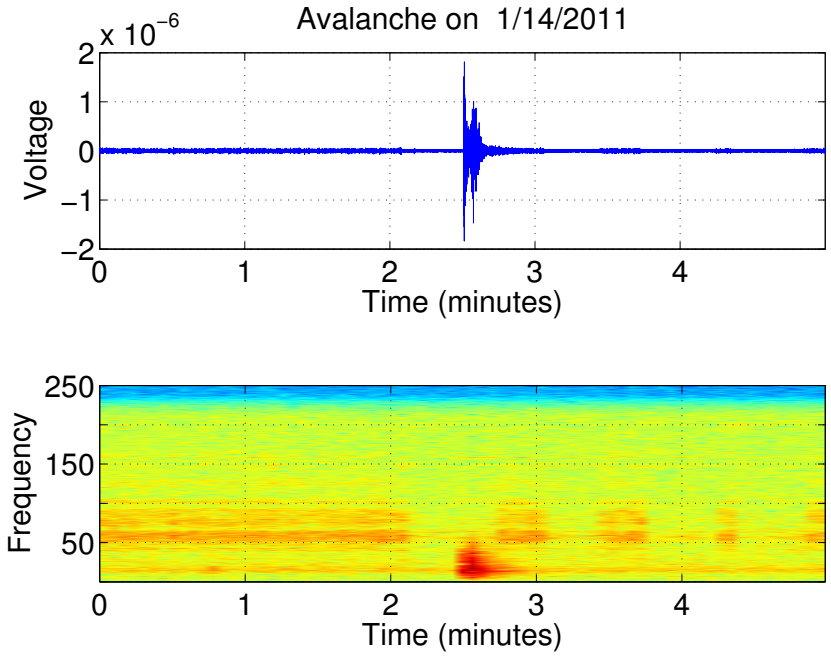


Figure 3.12: Five minutes of geophone data containing a large slab avalanche that occurred on January 14^{th}, 2011, plotted in both the time (top) and frequency (bottom) domains.

once at the beginning of each test, to simulate hardcoding Φ in Flash memory. For the other two algorithms, we allowed Φ (and the associated timing vector) to be regenerated after each N samples were collected and $y = \Phi\, x$ was calculated.

As described previously, we calculated the errors of the recovered versus original signals using the NRMSE metric. The results in Figure 3.13 show the mean NRMSE results with 95% confidence intervals for the three on-mote compressive sampling algorithms. The most notable trend in the NRMSE results is that all three algorithms performed about the same. In particular, our simplistic and lightweight RTV algorithm works as well as ARS and SBS for compressively sampling a real-world signal. We note, again, that the ARS and SBS algorithms are actually infeasible at this moderately high sampling rate (i.e., 500 Hz).

For our RTV algorithm, regenerating the timing vector V_t on-mote after each N samples is not realistic, since it takes 139.8 ms (on average) to generate a new timing vector. Thus, we evaluated the performance of RTV when we tested seven different frequencies with which

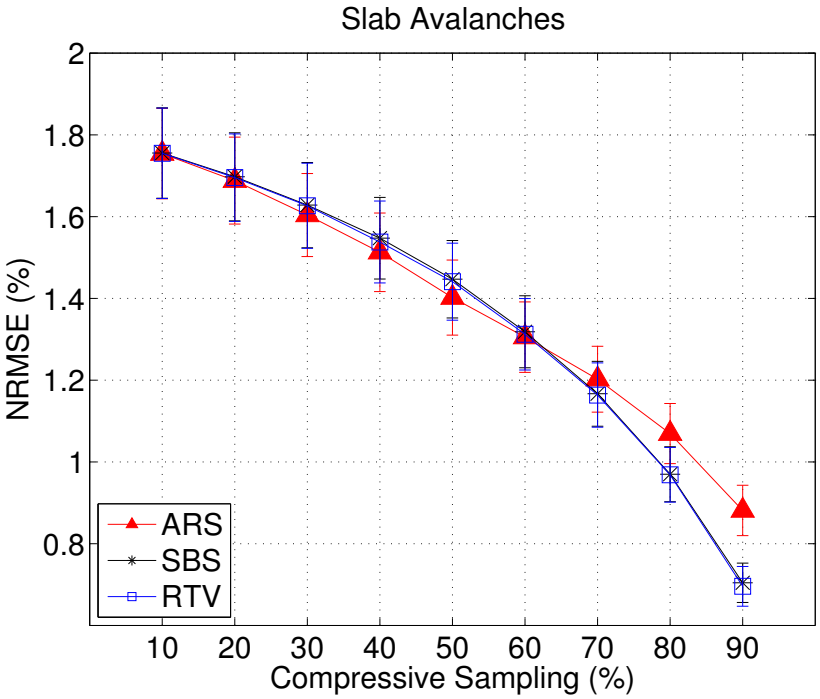


Figure 3.13: Mean NRMSE results with 95% confidence intervals from simulating three on-mote compressive sampling algorithms on the passive seismic data sampled at 500 Hz.

to regenerate V_t, i.e., always, every 30 seconds, 60 seconds, 90 seconds, 120 seconds, 150 seconds, and never. Specifically, the timing vector was regenerated after every $N = 500$ samples, $30N$, $60N$, $90N$, $120N$, $150N$, and never, respectively. In the "never" case, the timing vector was only generated once (at the beginning of the simulation). Our results indicate that all seven regeneration intervals were about the same (see Figure 3.14), with the percentage root-mean-square difference (PRD) of recovered signals falling well within every interval's 95% confidence intervals. PRD is defined as

$$PRD = \frac{||x - \widehat{x}||_2}{||x||_2} \times 100,$$

where x is the original signal, $\widehat{x}$ is the estimated recovery, and $||x||_2$ is the magnitude of vector x [34]. These results imply that we never have to modify the timing vector; in other words, the RTV algorithm can perform well even if we only calculate V_t once during mote initialization.

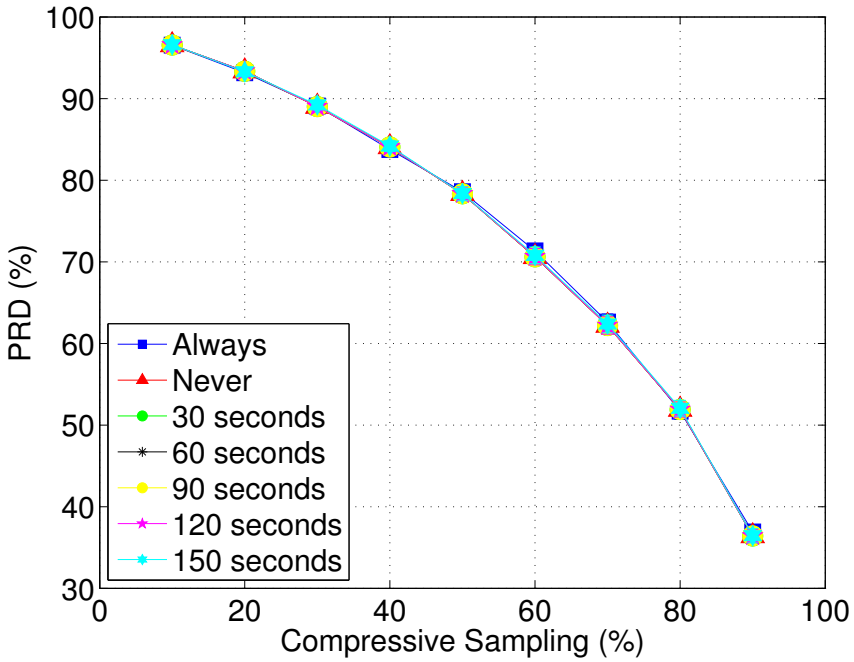


Figure 3.14: We experimented with recalculating the timing vector V_t at various intervals. We plot the percent root-mean-square difference (PRD) and 95% confidence intervals of the signals recovered using the seven different recalculation intervals.

Our results show that all three compressive sampling algorithms performed similarly when simulated on real-world passive seismic data. Furthermore, our simulations results show that the RTV algorithm performs about the same regardless of how often V_t is regenerated. Again, we note that the simulation results for ARS and SBS must be taken with a grain of salt, since both algorithms are infeasible at such moderately high sampling rates.

3.4 Conclusions

In this chapter, we present a new lightweight on-mote compressive sampling algorithm called RTV and show that it can significantly reduce power consumption in high sample rate wireless sensor applications. In addition, we show results that compare RTV to the two existing on-mote compressive sampling algorithms that exist in the literature, i.e., ARS and SBS. Our experimentation included three sets of tests: 1) recovering a simplistic sparse sinusoid produced by a signal generator, 2) analyzing the power consumption of each algorithm,

and 3) simulating the algorithms on real-world passive seismic data containing avalanches.

The results indicate that our RTV algorithm outperforms ARS and SBS in several ways. First, unlike ARS and SBS, RTV still functions properly at moderate to high sampling rates (e.g., 500 Hz or above). Second, RTV consumes the least amount of power, since it does not need to calculate the next random sleep time (as in ARS) or $y = \Phi x$ (as in SBS). Third, unlike SBS, which requires hardcoding a sparse representation of Φ in Flash memory to be feasible, the RTV algorithm has a significantly smaller memory footprint. Lastly, compared to ARS, the RTV algorithm does not introduce costly software based sampling jitter errors by constantly stopping, recalculating, resetting, and restarting timer interrupts.

Our RTV algorithm is ideal for applications that concern geohazard monitoring (e.g., avalanche, rockfall, or landslide detection) where geoscientists have some a priori knowledge of the signal they plan to acquire. An advantage of our RTV algorithm is that, instead of implementing some form of customized on-mote event detection, our RTV algorithm shifts the computational burden away from the wireless mote, which decreases power consumption and sensor node computational complexity. Furthermore, RTV is advantageous because it is non-adaptive, meaning that the rate of compression does not directly depend on signal redundancies; instead, compression rates, and thus, power consumption, can be guaranteed. In other words, with RTV, a wireless sensor system can be designed with user-specified compression rates and power consumption before deployment.

In the next chapter, we compare our RTV algorithm against other on-mote lossy compression techniques. Our intuition is that RTV will consume the least amount of power, since it decreases the number of analog to digital conversions and does not require any arithmetic after initialization.

CHAPTER 4

ON-MOTE LOSSY COMPRESSION ALGORITHMS

Creating a low-cost wireless sensor network (WSN) for continuous (e.g., 250 Hz sampling rate) geohazard monitoring necessitates a better approach than a simplistic "sense, send" modality. Since the radio on a wireless device consumes orders of magnitude more power than other components (e.g., ADC and CPU) [8], streaming all the data may consume too much power to be viable. As such, using compression to reduce radio transmissions will help increase system longevity, decrease overall system power requirements, and decrease system costs.

There have been several lossy compression[5] algorithms devised specifically for resource constrained wireless motes (e.g., 8-16 MHz CPU and 2-10 KB RAM). These algorithms include: K-run-length encoding (KRLE) [37], lightweight temporal compression (LTC) [38], wavelet quantization thresholding and RLE (WQTR) [39], low-pass filtered fast Fourier transform (FFT) [40], and compressive sampling (CS) [32, 34, 41]. In this chapter, we compare these five on-mote, lossy compression algorithms as potential data reduction techniques on real-world seismic data.

The main contribution of this chapter is a rigorous evaluation and analysis comparing our lightweight and novel CS technique called Randomized Timing Vector (RTV) [41] to four other on-mote, lossy compression algorithms (KRLE, LTC, WQTR, and FFT) using identical real-world seismic data sets. Consequently, this work provides a novel comparative study of five state of the art on-mote, lossy compression techniques. Previous literature comparing on-mote lossy compression algorithms [42] did not simulate, implement, or evaluate the compression algorithms on the *same set of data*. Instead, the authors of [42] discussed the merits of each algorithm based on the mutually exclusive results presented in the literature

[5]A lossy compression algorithm means that some information is lost during compression and decompression.

surveyed; in other words, the algorithms were compared based on results from different data sets. We also note that the survey conducted by [42] did not include KRLE, WQTR, FFT, or our CS algorithm, RTV.

Results depicted in this chapter demonstrate why CS, a lightweight and non-adaptive compression algorithm, is an attractive option for on-mote lossy compression. Specifically, CS offers guaranteed compression performance, low recovery error, and low power consumption without subjugating decompressed signal quality. We note that lossy compression is not appropriate for exploration geophysics, where little is known about the target signal being acquired. However, lossy compression is a feasible data reduction technique for seismic event detection, where there is a priori knowledge of the target signal and some information loss is acceptable. Moreover, we note that lossless compression is not covered in our evaluation; comparing lossy to lossless compression is beyond the scope of this dissertation.

4.1 Background

In this section, we describe the five lossy compression algorithms used in this study. We provide implementation details where appropriate.

4.1.1 K-Run-Length Encoding (KRLE)

The authors of [37] propose a novel lossy adaptive compression algorithm, called K-run-length encoding (KRLE), which allows for some variability in the input data stream during data encoding. KRLE encodes the input signal by using a range of acceptable values specified by a parameter K. Specifically, the current value (y) of an input stream is considered redundant if it falls within some predefined range of the first novel value (x), that is, if $x - K \leq y \leq x + K$. For example, with $K = 3$, the sequence of integers $\{61, 62, 63, 64, 65\}$ are encoded simply as $\{61 : 4; 65 : 1\}$, which indicates that the decoder should reconstruct the value 61 four times and then the value 65 once. Experimental results from [37] show that KRLE can significantly increase compression rates of certain signals, where the compression rate is defined as:

$$CompressionRate = 100 \times \left(1 - \frac{CompressedSize}{OriginalSize}\right).$$

4.1.2 Lightweight Temporal Compression (LTC)

Much like KRLE, lightweight temporal compression (LTC) adaptively compresses data by encoding streams of redundant sequences [38]. LTC is different from KRLE in the way redundancy is defined. In LTC, a data point is considered redundant if it falls within some range of lines interpolated from previous data points. If the current data point falls within some user-specified range of interpolated lines (specified by a parameter K), then the data point is encoded as redundant. Otherwise, the current data point is used to start the next iteration of interpolation and compression. Results from [38] show that LTC performs comparably to Lempel-Ziv-Welch and wavelet based compression on micro-climate data.

4.1.3 Wavelet Quantization Thresholding and RLE (WQTR)

The authors of [39] describe a lossy adaptive compression algorithm that we refer to as wavelet quantization thresholding and RLE[6] (WQTR). First, the WQTR algorithm works by calculating a discrete wavelet transform using Cohen-Daubechies-Feauveau (2,2) integer wavelets (CDF(2,2)) on subsets of 128 samples. Integer wavelets were selected because they can be implemented using only addition and bit shifting operations. Second, the wavelet coefficients are quantized to reduce signal resolution, which decreases the size of the signal and makes the signal more compressible. Third, the coefficients undergo thresholding, where coefficients with absolute values above some percentage threshold are kept while the other coefficients are zeroed out. The resulting signal, consisting of a few large quantized wavelet coefficients and many zeros, is then passed to a run-length-encoder (RLE) and transmitted. Results from [39] show that WQTR's increased compression rates and low overhead make it a viable option for the authors' WSN deployment.

[6]RLE, which stands for Run-Length Encoding, is equivalent to KRLE with $K = 0$.

4.1.4 Low-pass Filtered Fast Fourier Transform (FFT)

The fast Fourier transform (FFT) is a well-known and efficient method to transform signals from the time to frequency domain [40, 43]. Briefly, an N-point FFT takes N complex numbers as input and produces N complex FFT coefficients; within a mote's memory, the FFT's input and output both consist of N real and N imaginary components. Assuming the imaginary component of the input is zero, as it is with a real-valued seismic signal, the real and imaginary components of an N-point FFT's output are symmetric and antisymmetric about the center frequency, respectively. Thus, instead of transmitting $2N$ numbers to represent the FFT's N complex coefficients, we make use of the FFT's symmetry to transmit $N/2$ real and $N/2$ imaginary components (i.e., the first "half" of the FFT's output). We recover the N complex FFT coefficients by mirroring the real and imaginary coefficients about the center frequency and multiplying the mirrored imaginary components by -1 (since it is antisymmetric).

To implement non-adaptive compression, we employ low-pass filtering on the Fourier coefficients before transmission; low-pass filtering allows low-frequency coefficients to "pass-through" the filter, zeroing out the frequency coefficients above a user-defined threshold [40]. In other words, we achieve non-adaptive compression by transmitting the lowest (in terms of frequency bins) L real and L imaginary components of the FFT's output, where $L < N/2$. During offline signal recovery, the $N/2-L$ real and $N/2-L$ imaginary components not transmitted are set to 0. The full time-domain signal is recovered using the inverse Fourier transform on the N recovered complex FFT coefficients. Our implementation of FFT compression is based on the source code provided by the Open Music Labs [43], which computes a fixed point FFT and is written in assembly code for fast computation. We note that, although FFT-based adaptive compression (similar to WQTR) could have been implemented, we chose the non-adaptive approach for the sake of comparison against CS, which is a non-adaptive compression algorithm.

4.1.5 Compressive Sampling (CS)

Compressive sampling (CS) is the final lossy compression algorithm evaluated in this chapter. While other on-mote CS algorithms such as Additive Random Sampling [32] and Sparse Binary Sampling [34] have been proposed, herein we use our Randomized Timing Vector (RTV) algorithm due to its superior performance (as discussed in Chapter 3 and [41]).

CS is advantageous because it greatly simplifies the compression process by acquiring compressed signals directly and shifts the computational burden away from the wireless mote to the system performing signal decompression. Thus, given the lightweight and non-adaptive nature of CS data compression, how does CS compare to the four other lossy algorithms (three adaptive and one non-adaptive) described previously? To answer this question, we analyzed the performance of the five lossy compression algorithms in two scenarios: 1) in simulation on two sets of real-world seismic data and 2) on real hardware. We describe our experimentation, implementations, and results in the next two sections.

4.2 Simulation

We first simulated the five lossy compression algorithms on real-world seismic data collected in the mountains above Davos, Switzerland. The simulation experiments allowed us to evaluate the compression algorithms in terms of compression rates, recovery error rates, and event classification accuracies. Additionally, we further support our findings by simulating the compression algorithms on a second real-world seismic data set collected from a test levee in the Netherlands called IJkdijk (pronounced "Ike-dike"). In total, we simulated on-mote lossy compression on nearly 3.58 hours of real-world seismic data.

The seismic data used in the first set of simulations was collected during the 2009-2010 winter season using a *wired* geophone array in the mountains above Davos, Switzerland (see Section 2.2 and [9] for more details). The full seismic data set, sampled at 500 Hz with 24-bit precision, contains over 100 days of data with 33 slab avalanche events (e.g.,

Figure 4.1). In the experiments presented in this section, we focus on subsets of seismic data containing the 33 slab avalanches only; this data subset was selected to evaluate the compression performance on our target signals, i.e., avalanche events. In Section 4.4, we expand our analysis to include other types of seismic data (e.g., high energy noise and "silence").

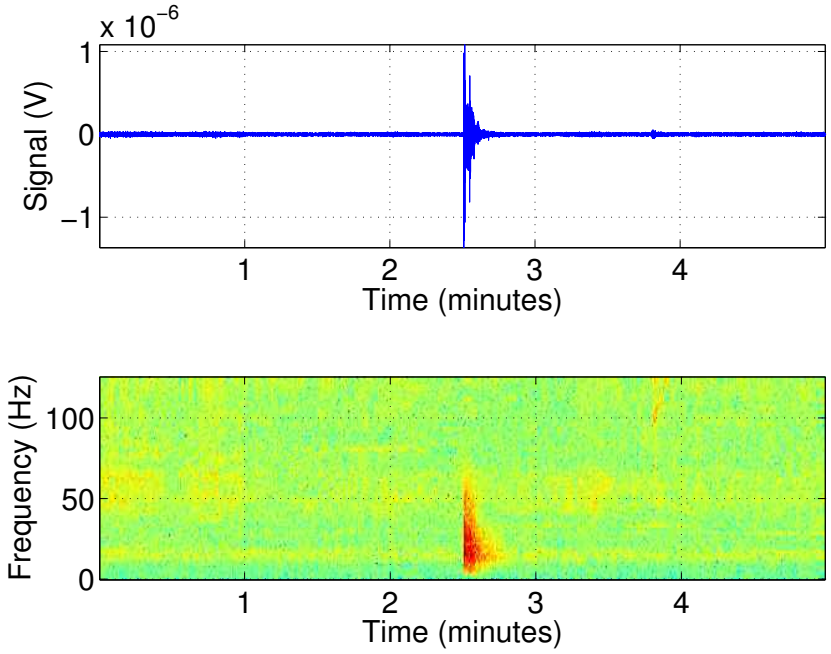


Figure 4.1: Five minutes of seismic data containing a slab avalanche event, plotted in the time domain (top) and time-frequency domain (bottom). We simulated on-mote lossy compression on 2.75 hours of real-world seismic data, which contained 33 slab avalanche events that occurred in the 100+ days of data.

To mimic the hardware limitations at the time of experimentation, we subsampled the 33 slab avalanches from the original 500 Hz sample rate with 24-bit precision to a 250 Hz sample rate with 16-bit precision. At the time of experimentation, our custom wireless motes could not sample continuously (and provide a stream of data) faster than 250 Hz sampling.[7] Additionally, we performed compression on the raw ADC encodings, as these values are two byte signed integers, not floating point voltage values.

[7]Current custom mote prototypes can sample much faster; see Chapter 5 for details.

In terms of algorithm parameters, we selected powers of two for KRLE and LTC (i.e., $K = \{1, 2, 4, ..., 512\}$). For WQTR, we used thresholds between 10% and 90%, and added a 98% threshold for aggressive compression. For FFT, we selected low-pass filter thresholds between 10% and 90% of the center frequency. For CS, we employed compressed vector lengths of $M = \{0.1N, 0.2N, ..., 0.9N\}$. Note that the ratio of compressed vector length to full signal (i.e., M/N) is inversely proportional to the compression rate, e.g., a 30% ratio between M and N results in a 70% compression rate.

To increase the credibility of our simulations, we used the same C++ compression functions and memory usage as our on-mote implementations of CS, KRLE, LTC, and WQTR (see Section 4.3). In other words, we first implemented the compression functions in C++ for Arduino and then ported the methods to a desktop computer using a different driver to read in the seismic data. Additionally, we simulated the memory constraints of the Arduino Fio platform by limiting the size of each data buffer to be compressed. We note that since the FFT library obtained from the Open Music Labs [43] was written in assembly for Arduino, we used custom Matlab functions (with limited precision) for our simulations.

To simulate our hardware implementation, we compressed signals using a two buffer method; while one buffer of data was being acquired, the other buffer was being compressed and transmitted. Specifically, for KRLE, LTC, and CS, we compressed buffers of $N = 256$ short (two byte) integers at a time. Due to the increased memory required for computation, we compressed buffers of $N = 128$ short integers for FFT and WQTR.

In terms of radio usage, we simulated binary transmissions. For CS, this meant transmitting the M short integers selected during compression. For KRLE and LTC, we transmitted three bytes at a time: two bytes for the signed short integer and one byte for the number of occurrences. For FFT, we transmitted $2L$ short integers corresponding to the L real and L imaginary components of the low-pass filtered FFT coefficients. Lastly, for WQTR, we transmitted five bytes at a time: a four-byte floating point wavelet coefficient followed by one byte for the number of occurrences. We omitted the quantization step of WQTR due to

extremely poor performance during simulation; though quantization equates to better compression rates, the loss of precision from normalizing, truncating, and/or interpolating the wavelet coefficients from four-byte floating points to two-byte shorts resulted in significantly higher recovered signal error rates. In other words, quantizing the wavelet coefficients to match the other algorithms (i.e., from floating point to short integer) resulted in recovered signals that were unusable.

4.2.1 Compression Rates

After simulating compression on the 33 five-minute chunks of data containing a slab avalanche, we calculated the compression rates using the formula defined in Section 4.1.1. Figure 4.2 shows histograms of the compression rates for each algorithm over the entire simulation.

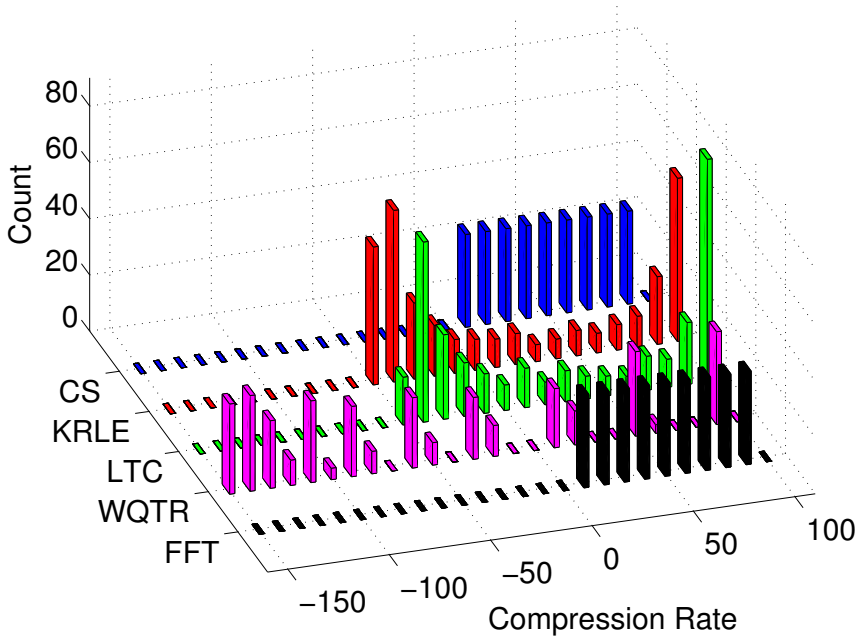


Figure 4.2: Histograms of the compression rates from the simulation on the avalanche data. Note that CS and FFT (the two non-adaptive algorithms) are the only algorithms without negative compression rates.

The compression rates presented in Figure 4.2 show two clear advantages to using non-adaptive compression. First, nonnegative compression rates[8] can be guaranteed. In other words, the rate of compression for non-adaptive algorithms does not depend on the compressibility of the input signal. Additionally, with CS and FFT, users can specify a compression rate for the lifetime of the mote by selecting the M parameter and the low-pass filter threshold, respectively.

The second advantage is that the non-adaptive compression algorithms have less variability in the resulting compression rates. We encourage the reader to note the high variability of compression rates for KRLE, LTC, and WQTR in Figure 4.2 compared to the low variability of compression rates for CS and FFT. In the case of KRLE and LTC, the combination of small K values and high signal variance led to negative compression rates.[9]

4.2.2 Recovered Signal Error

For signal recovery (decompression), we utilized a combination of C++, Python, and Matlab. Decoding KRLE and LTC based encodings was straightforward, since these algorithms specify what number to print and how many times it occurred. WQTR decompression also included this decoding step, followed by an inverse wavelet transform. Decoding the FFT algorithm's output required recovering the N FFT coefficients and computing the inverse FFT. Lastly, for CS, we employed reweighted ℓ_1-norm (RWL1) minimization [44] assuming sparsity in the time-frequency domain (Gabor atoms); in Appendix B, we show that RWL1, assuming sparsity in the time-frequency domain, is a good choice for our real-world seismic data from avalanche monitoring.

We determined signal recovery errors by computing the normalized root mean square error (NRMSE) for the decompressed versus original signals. See Section 3.3.1 for details regarding how NRMSE is calculated. Figure 4.3 plots the mean NRMSE and 95% confidence

[8]Negative compression rates occur when the number of bytes required for the compressed signal encoding is greater than the original signal.

[9]For example, with a K value of 1, KRLE would encode a signal {0,2,4} as {0:1,2:1,4:1}, which requires more bytes than the original signal.

intervals of all five algorithms based on their respective mean compression rates. For KRLE, LTC, and WQTR in Figure 4.3, the K values and thresholds increase from left to right. For example, KRLE with $K = 1$ resulted in approximately -47% mean compression rate and almost zero NRMSE, while $K = 512$ resulted in 97% mean compression rate and 0.017 NRMSE.

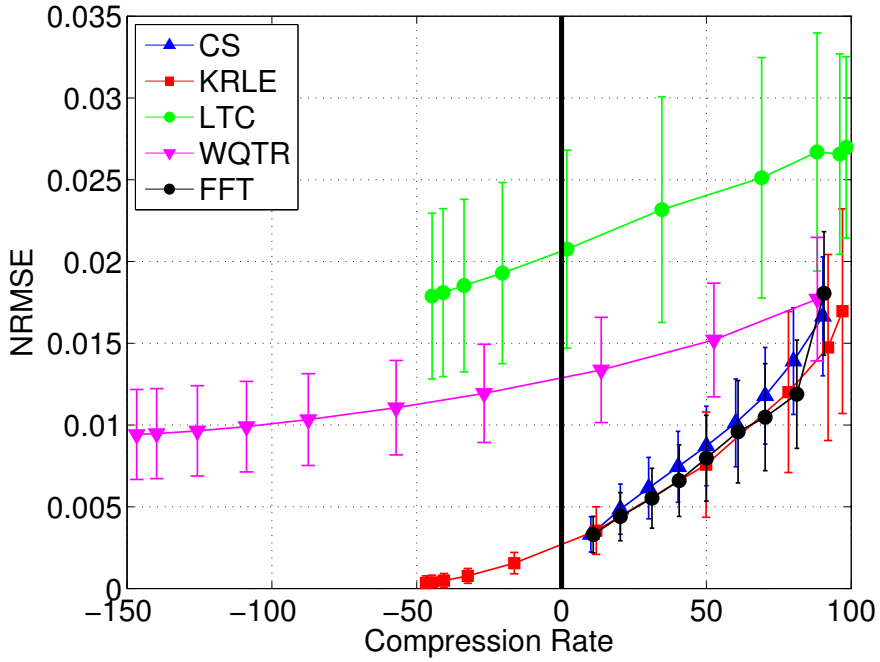


Figure 4.3: Mean NRMSE results (with 95% confidence intervals) from five lossy wireless node compression algorithms simulated on a real-world seismic data set containing avalanches.

The NRMSE results in Figure 4.3 show that CS, a lightweight non-adaptive compression technique, performs as well as the best performing algorithms: KRLE and FFT. In other words, the recovery errors of signals compressed and decompressed with CS fall within all the 95% confidence intervals of KRLE and FFT. Moreover, as shown, KRLE sometimes provides negative compression rates.

The NRMSE results are quite striking, considering that CS compression does *not* necessitate acquiring and storing every sample in the original signal. In other words, instead

of acquiring the full signal and *then* performing compression, CS allows us to acquire the compressed signal directly. In geophysical monitoring applications where data acquisition is expensive (e.g., due to power consumption from high sampling rates), CS is an attractive option because it is the only compression technique that does not require acquiring the full signal first (see our work in Chapter 3 and [41] for more details).

Though NRMSE results provide a nice depiction of compression rates and recovery error, the error rates alone do not paint a complete picture. For example, what does it mean for CS to have 0.017 mean NRMSE at 90% compression rate? Is the recovered signal still useful?

4.2.3 Avalanche Event Classification

In hopes of answering such questions, we applied our automated avalanche detection workflow from Chapter 2 and [45] to the recovered (decompressed) signals from the five compression algorithms. Specifically, for each of the recovered and original signals, we extracted features before training and testing a decision tree classifier using a 10-fold cross-validation procedure. We used stratified subsampling to create the training and testing subsets, which included all five-second avalanche frames and an equal number of randomly selected non-avalanche frames (to avoid overfitting).

We ran a 10-fold cross-validation procedure 100 times per recovered signal, randomly selecting a new set of non-avalanche frames each time. Figure 4.4 plots the mean classification accuracies and 95% confidence intervals of the five evaluated compression algorithms based on mean compression rates.

There are three interesting trends in Figure 4.4 worth noting. First, classification results show that CS performed quite well. For example, the mean accuracy of detecting slab avalanche events recovered from 60% compressive sampling (40% compression rate) was 91.3%. In comparison, with full sampling, we reached 92.3% mean classification accuracy. In other words, the 40% increase in compression rate for CS over full sampling resulted in only a 1% decrease in mean classification accuracy.

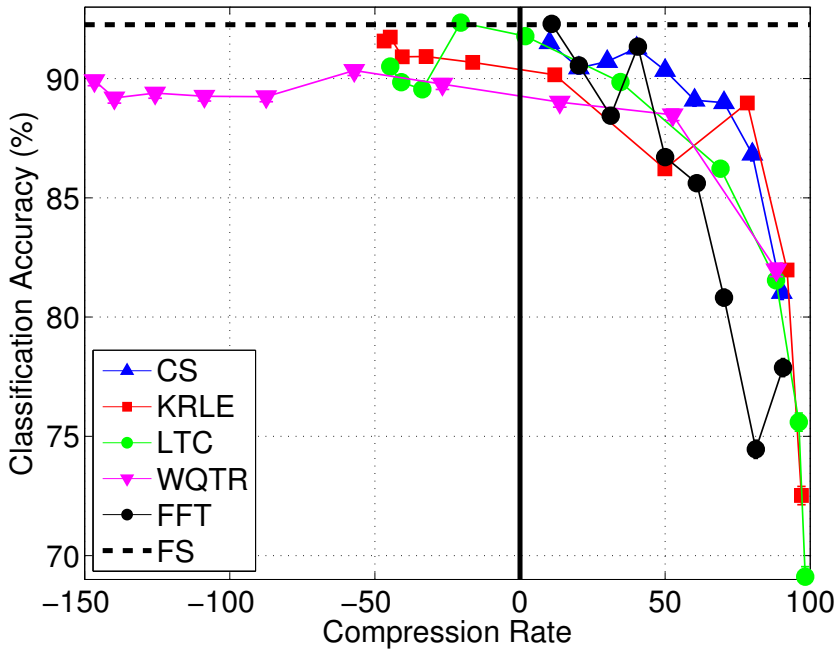


Figure 4.4: Mean and 95% confidence intervals of classification accuracies from our machine learning workflow (to detect slab avalanches) performed on the recovered signals. FS shows the classification accuracy for the full (uncompressed) signal.

Second, observe the "bumps" of increased classification accuracies for KRLE with parameter $K = 128$ (78% compression rate) and FFT with low-pass filter thresholds of 60% and 10% (40% and 90% compression rates, respectively). We believe that these temporary improvements in classification accuracies occur because the algorithms were effectively denoising the data before feature extraction. For example, by encoding many values as a single number, KRLE removes frequency content and greatly simplifies the signal during periods of little variance. However, the classification accuracies of KRLE with $K = 256$ and $K = 512$ (92% and 97% compression rates, respectively) degrade rapidly to 82% and 73%, respectively. Likewise, the FFT's mean classification accuracies quickly degrade from above 90% accuracy (with 40% compression rate) to below 75% accuracy (with 80% compression rate). This rapid downward trend in mean classification accuracies given higher K values and lower filter thresholds suggests that both KRLE and FFT remove useful frequency

information from the signal before feature extraction and machine learning.

Lastly, although CS and FFT both had exclusively nonnegative compression rates, FFT's accuracies were highly variable or significantly worse than the other algorithms evaluated. We hypothesize that this occurs because the low-pass filtered FFT explicitly removes the mid to high frequency components of the recovered signal, thus eliminating information that may be critical for our pattern recognition workflow to detect avalanches. Herein lies an advantage of CS over FFT; with high rates of compression (i.e., greater than 50%) it appears that CS recovers more useful information from the compressed signal than the low-pass filtered FFT. For example, with 50% compression rate, FFT had a mean classification accuracy of 86.7% while CS had a mean classification accuracy of 90.3%.

4.2.4 IJkdijk Seismic Data

Given that CS compares favorably to other lossy compression algorithms in terms of compression rates, NRMSE, and classification accuracies on one data set, how does CS perform on a different data set? To answer this question, we ran all five compression algorithms on a seismic data set collected from the IJkdijk test levee in the Netherlands. Briefly, IJkdijk is a test levee that is monitored and measured in various ways while it is brought to failure. Geoscientists collected several days of 16-bit passive seismic data from 24 *wired* geophones as the levee was brought close to failure. For our experiments, we simulated compression on a small segment of data (about 50 minutes) from a single geophone sensor deemed interesting by the team of geophysicists, geologists, and geotechnical engineers (e.g., Figure 4.5). The data was subsampled from 4000 Hz to 250 Hz to mimic the current bandwidth limits of our low cost *wireless* geophysical sensors. Results from our compression simulations are plotted in Figures 4.6 and 4.7, which show the compression rate histograms and the mean NRMSE with 95% confidence intervals for the recovered data, respectively.

Though the errors in Figure 4.7 are larger than in Figure 4.3, the relative ordering of the lossy compression algorithms remains approximately the same. In this case, with 50% or less compression rates, CS performs the best; above 50% compression rates, CS was tied with

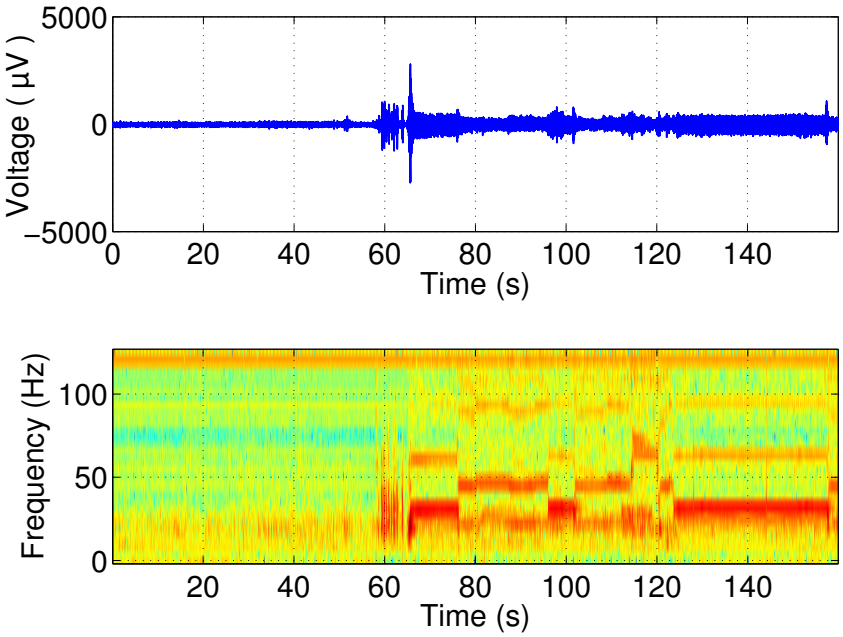


Figure 4.5: We simulated the five lossy compression algorithms on a second real-world seismic data set collected from the IJkdijk test levee in the Netherlands. We plot example data in the time domain (top) and in the time-frequency domain (bottom).

FFT as the best performing algorithm. It is interesting to note the relatively flat performance of WQTR (in terms of NRMSE). We hypothesize that this is due to the simplicity of the CDF(2,2) integer wavelet and relatively small buffer size used in WQTR compression.

We did not evaluate the IJkdijk signals in terms of machine learning accuracy, simply because we have not yet designed a pattern recognition workflow to detect stages of levee failure (e.g., seepage, erosion, and collapse). Such a workflow would help us evaluate the relative performance of the five compression algorithms in regards to whether the recovered signals are really useful. We hypothesize that the recovered signal for CS is useful, however, as the NRMSE is relatively low.

Another aspect worth noting between Figures 4.3 and 4.7 is that the mean compression rates of two adaptive compression algorithms (KRLE and LTC) differed between signals. Note how much the mean compression rate (x axis) decreases between the avalanche and

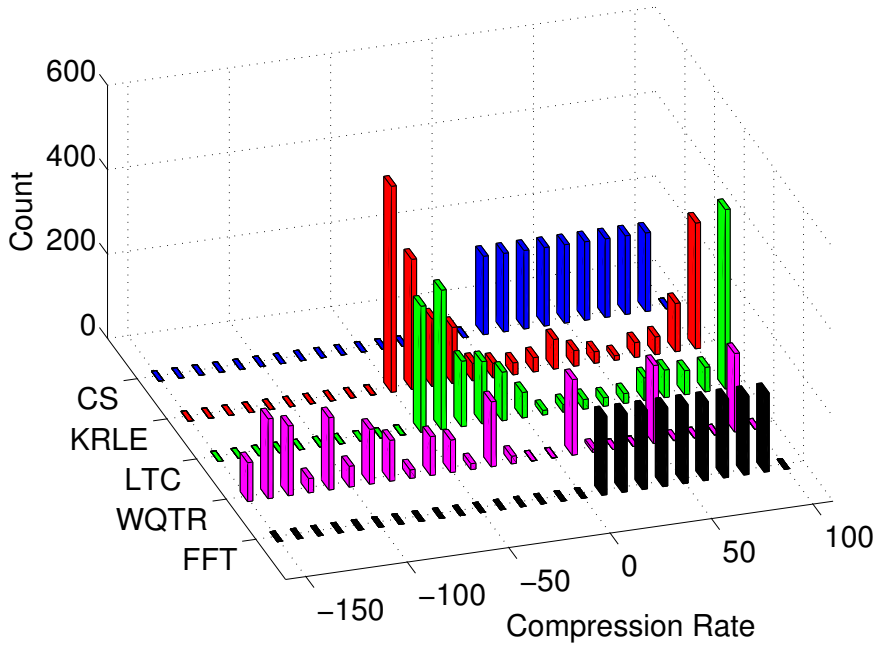


Figure 4.6: Histograms of the compression rates from the simulation on the IJkdijk data.

IJkdijk data sets; for example, the mean compression rate of KRLE with $K = 64$ (fourth from the right) decreased from 49.9% in the avalanche data set to 19.5% in the IJkdijk data set. Additionally, the mean compression rates for LTC decreased as well. In comparison, compression rates for FFT and CS did not change between data sets. In other words, once we pick M for CS and the low-pass frequency threshold for FFT, we can calculate what the compression rates will be. These results demonstrate how compression rates for some adaptive compression algorithms depend ultimately on the signals being compressed.

In short, these results illustrate an advantage for the non-adaptive algorithms over the adaptive ones; with FFT and CS, users can specify compression rates that will be guaranteed for the lifetime of the wireless mote. For KRLE, the most comparable performing lossy algorithm evaluated, users must select a value for parameter K and hope that compression rates will be nonnegative. Additionally, users must guard against selecting a K value that is "too big" for the target signal, which would lead to lost signal information.

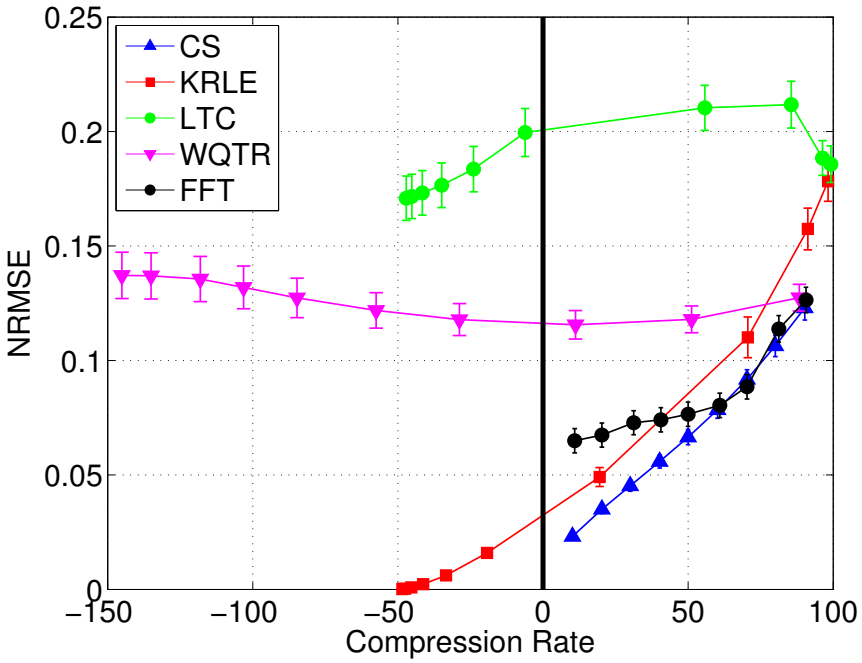


Figure 4.7: Mean NRMSE results (with 95% confidence intervals) from five lossy compression algorithms simulated on the real-world IJkdijk seismic data set.

Of course, as with the other parameterized lossy compression algorithms evaluated, both FFT and CS suffer from the same challenge: i.e., how to best select the ideal parameter to provide maximal compression without subjugating recovered signal quality. Despite the challenge of parameter selection, we argue that the non-adaptive algorithms are preferable to the adaptive algorithms because of the ability to precisely estimate compression rates, radio usage, and thus, power consumption. For the adaptive algorithms, how would users estimate the mote's power requirements based on selecting, for example, $K = 64$ versus $K = 128$? With CS and FFT, users can estimate power requirements and thus, system cost, with a very high degree of confidence. In other words, CS and FFT users do not have to pad power requirements to guard against extra power consumption that occurs from possible negative compression rates. We investigate this issue in detail in the next section.

4.3 Hardware Implementation

To further evaluate the five lossy compression methods, we implemented the algorithms on a low-cost Arduino Fio wireless mote platform (i.e., 2 KB RAM and 8 MHz CPU) with a long-range XBee Pro 802.15.4 radio module. Arduino Fio is our mote platform of choice because we have built high precision geophysical sensing "shields" that can plug and play with these low cost and easy-to-use platforms (see Chapter 5). Additionally, we used high power (1.5 km line-of-sight) radios to mimic a real-world wireless sensor deployment on a typical avalanche path or earth dam.

For repeatability, we tested the algorithms by compressing 36 seconds of real-world seismic data hard-coded in the mote's Flash memory. For our experiments, the 36 second test signal was synthesized from three short and low-noise slab avalanche events from the Swiss data set (see Figure 4.8).

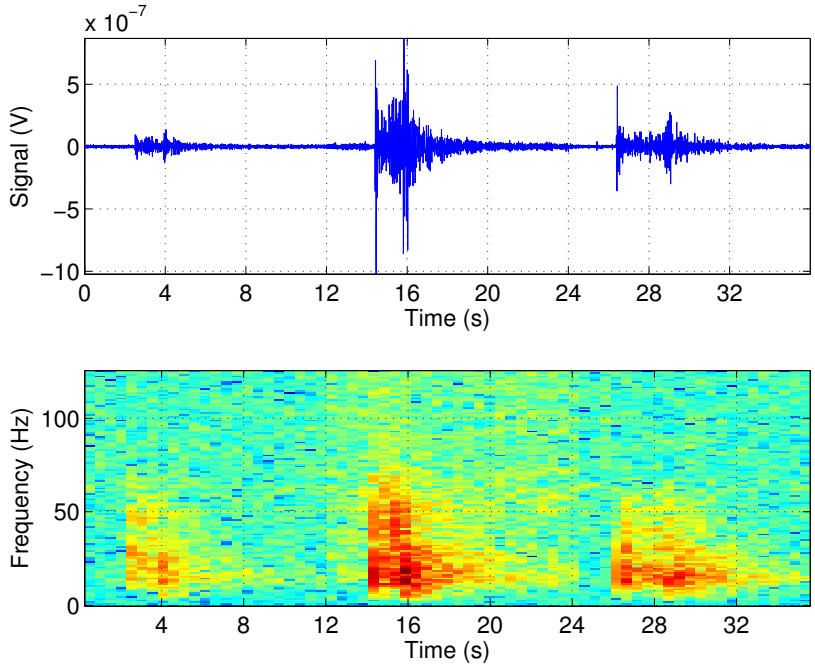


Figure 4.8: For repeatability, we stored 36 seconds of synthesized seismic data in Flash memory. The 36 seconds of data contain three slab avalanches. In the time domain (top) and time-frequency domain (bottom) figures, the avalanches begin at approximately three seconds, 14 seconds, and 26 seconds, respectively.

Similar to our simulation experiments in Section 4.2, our hardware experimentation consisted of adjusting the compression algorithm parameters and evaluating the resulting compression rate, signal recovery error, and power consumption. As discussed in Section 4.2, we implemented a two-buffer modality for all algorithms (including CS); while one buffer was being acquired from Flash memory, the other buffer was getting compressed and transmitted. To minimize radio power consumption, we put the XBee Pro radio in a low-power sleep state and further buffered the transmissions into 100-byte payloads, the maximum size of the XBee Pro radio's packet payload. When the 100-byte payload buffer became full, we woke up the radio, transmitted the payload, and put the XBee Pro back to sleep.

4.3.1 Recovered Signal Errors

We compressed the 36 second test signal stored in Flash using each of the five algorithms with the same parameters as our simulations; see Section 4.2 for algorithm parameter values. Compression rates were calculated based on the size, in bytes, of the compressed versus original signals received. Signal recovery was performed offline with a combination of Python, C++, and Matlab using techniques summarized in Section 4.2.2. The solid shapes in Figure 4.9 plot the NRMSE versus mean compression rate of the five parameterized algorithms executed in hardware on the real-world test signal. The white-filled shapes in Figure 4.9 represent the NRMSE from simulated compression on the exact same 36 seconds of data.

The most notable trend in Figure 4.9 is that, for all but FFT, the NRMSE rates for signals recovered from our hardware implementations (solid shapes) and simulated compression (white-filled shapes) were identical. These results help validate the credibility of our simulation experiments in Section 4.2 by showing that a signal compressed in hardware is equivalent to the same signal being compressed in simulation. Despite our efforts to simulate the mote's limited precision for computing the FFT, there were small differences in the NRMSE results for the simulated versus on-mote FFT compression. We hypothesize that the simulated FFT performs better because it was implemented in Matlab on a 64-bit computer (with 64-bit computation); the on-mote version, on the other hand, was implemented

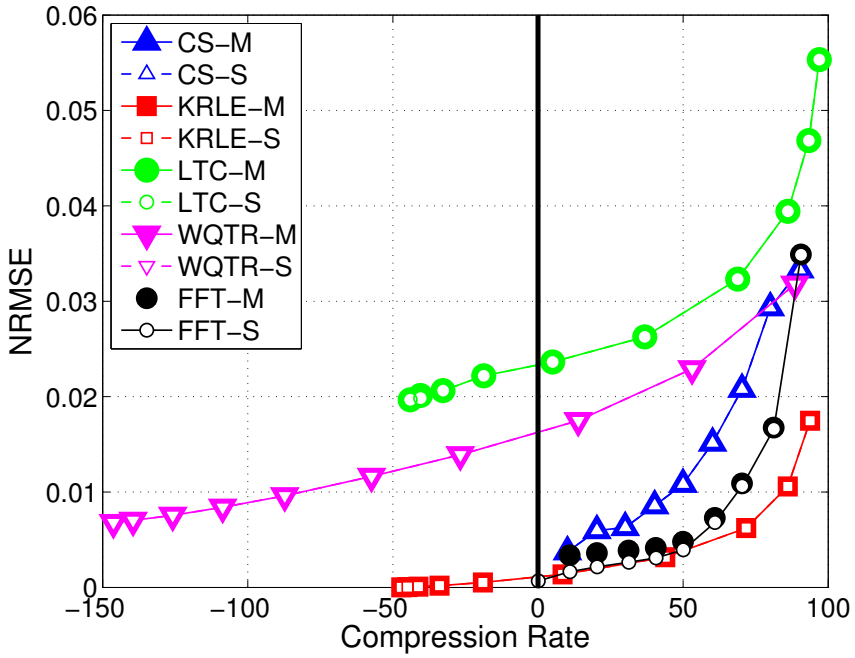


Figure 4.9: The NRMSE of the recovered signal compressed on mote (solid shapes) and in simulation (white-filled shapes) on 36 seconds of seismic data containing three slab avalanches.

in assembly on an 8-bit microcontroller.

Furthermore, because we simulated compression on a sizable 3.58 hours of seismic data in Section 4.2, the NRMSE results presented in Figure 4.9 should be observed with caution. In other words, it would be naive to conclude that KRLE is the best performing lossy compression algorithm for seismic data (in terms of NRMSE) due to the small, low variability 36-second data set used for the simulation results presented in Figure 4.9. Instead, we refer the reader to Figures 4.3 and 4.7, which show results from 2.75 hours and 50 minutes of seismic data, respectively; as shown in these two figures, CS was either the best or tied for the best performing algorithm (in terms of lowest NRMSE). We hypothesize that KRLE performed best on the small 36-second data set because the signal contained very little variability and noise events. This lack of high signal variability is unlike the large 2.75 hour data set used in simulation, which contains background noise events caused by helicopters,

airplanes, wind, ski lifts, etc.

4.3.2 Power Analysis

Lastly, we analyzed the power consumption of the wireless mote as it executed the five compression algorithms and compared the results to full sampling. Specifically, we measured the voltage difference across a 10.1Ω resistor in series from the mote to ground, then derived the current draw using Ohm's law. All voltages were measured at 50 KHz sampling rate using a 16-bit precision National Instruments USB-6218 DAQ and LabView SignalExpress. To reiterate, we used an Arduino Fio wireless mote with a long-range XBee Pro 802.15.4 radio module (1.5 km line-of-sight range). From our power analysis, we then estimated the longevity of a reasonably sized battery used to power an Arduino Fio wireless mote running these algorithms. Figure 4.10 depicts the estimated longevity of a 6.6 Ah battery in ideal conditions. The dashed line shows the "benchmark" battery life of full sampling (i.e., the original signal with 0% compression rate).

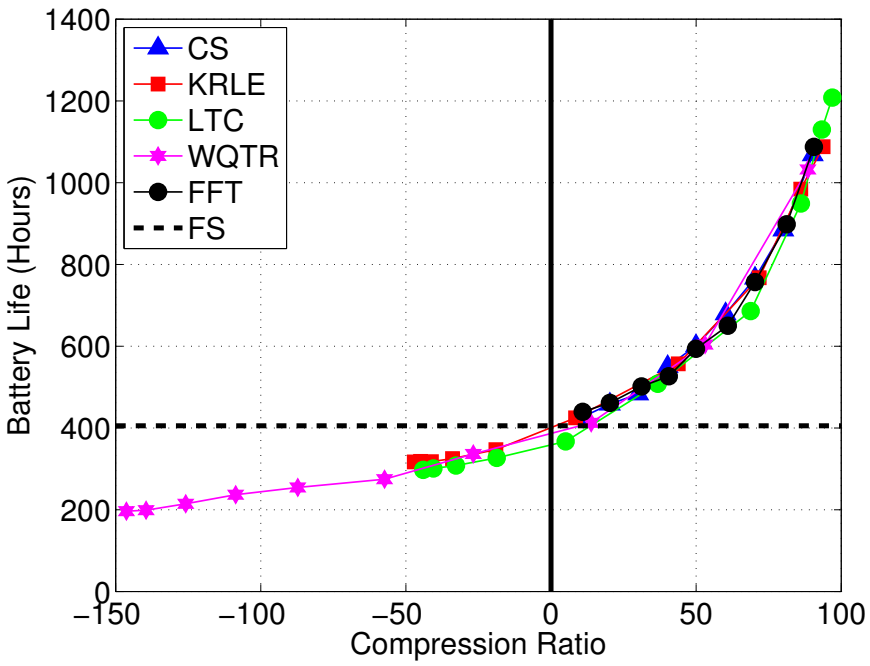


Figure 4.10: The estimated longevity of a 6.6 Ah battery used to power a wireless mote running each algorithm.

The most notable and conclusive trend in Figure 4.10 is how costly negative compression can be. Adaptive algorithms KRLE, LTC, and WQTR all had instances where negative compression rates resulted in power consumption greater than that of full sampling. In other words, without careful parameter selection in relation to the target signal, KRLE, LTC, and WQTR may drain the battery faster than full sampling. On a different note, the non-adaptive compression algorithms (i.e., CS and FFT) were the only methods with battery longevities exclusively above full sampling. Figure 4.10 illustrates, once again, the main advantage of CS and FFT, i.e., nonnegative compression rates and resultant power savings can be guaranteed.

Although CS and FFT both showed exclusively nonnegative compression rates, CS improves upon FFT by performing better in terms of avalanche event classification accuracies (Figure 4.4), implying that CS recovers more useful information in the decompressed signal. Moreover, CS is advantageous because it can be implemented without sampling the entire (full) signal before compression; such an approach is particularly useful when data acquisition is expensive (e.g., powered sensor or power hungry ADC).

4.4 Extended Analysis

In this section we describe results from extending our simulation and on-mote evaluation of the five on-mote lossy compression algorithms. The extended analysis is motivated by two factors: 1) to evaluate how the lossy compression algorithms perform on more diverse data and 2) to analyze the on-mote runtime characteristics of each algorithm. In particular, we first analyze experimental results from simulated lossy compression on nearly 38 additional hours of Swiss data, including other types of non-avalanche events and a full day of avalanche data. For the on-mote runtime evaluation, we recorded coarse and fine-grain runtime information of the algorithms executing on the Arduino Fio mote.

4.4.1 Simulation

The previous simulation results were based on only 2.75 hours of seismic data. In this section, we extend our evaluation to include more diverse signals, i.e., 38 additional hours of Swiss data. In the entire (100+ day) Swiss seismic dataset, the avalanches only account for about 0.02% of the data. The 99.98% remaining data is full of periods of low energy "silence" and non-avalanche seismic noise events from planes, helicopters, ski lifts, etc. In this section, we simulate the five lossy compression algorithms on six subsets of the avalanche data: five sets of non-avalanche "events" selected based on their root-mean-square (RMS) energy statistics, and an entire day of seismic data (i.e., April 24^{th}, 2010). Our extended analysis corresponds to nearly 38 hours of additional data and helps determine how well the lossy compression algorithms perform on more diverse data (i.e., not just slab avalanche events).

Root-mean-square (RMS) is a measure of signal energy. RMS is calculated as follows:

$$RMS = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}},$$

where n is the length of the original signal x. Using data from each of the 35 days that *do not* contain avalanche events, we calculated the RMS value for each five second frame of seismic data. In other words, for each day (about 24 hours) not containing an avalanche, we calculated roughly 17,000 RMS data points, one for each five second data window.

From this set of over 17,000 RMS values per day, we calculated the min, max, median, "closest to mean", and mode RMS values. We then extracted "events" based on these five RMS values; in other words, each non-avalanche day was processed into five representative events: a min RMS event, max RMS event, "closest to mean" RMS event, median RMS event, and mode (based on binning) RMS event. (Further details on how we extracted these five events follow). We selected non-avalanche days to eliminate overlap with previous results (e.g., a maximum RMS event might be a slab avalanche).

Each of the extracted event subsets were five minutes in length, with the corresponding five-second window (used for RMS calculation) located in the center (i.e., at 2.5 minutes). Thus, for each of the 35 days, we selected five RMS events, i.e., min, max, "closest to mean", median, and mode, each of which were five minutes in length. In total, this corresponds to an additional 14.5 hours of data to simulate.

Figures 4.11, 4.12, and 4.13 show how RMS energy was used to select events of interest. Our example data, shown in Figure 4.11, is from non-avalanche day 13 (February 23^{rd}, 2010). First, RMS energy is calculated for each five-second window (e.g., Figure 4.12) of this February day of data without an avalanche. Next, statistics of the RMS energy values are calculated and used to locate events of interest (e.g., Figure 4.13 and Table 4.1).

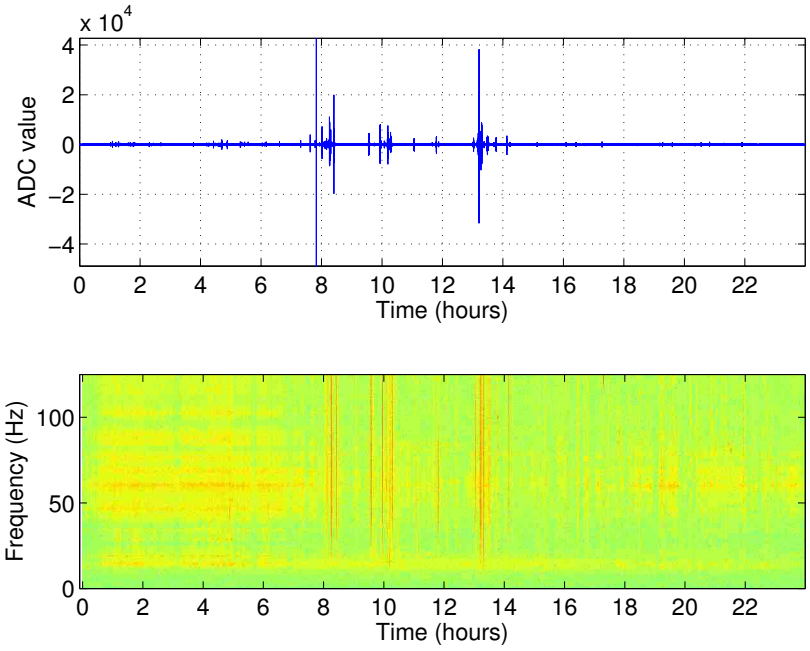


Figure 4.11: Data from day 13 of the Swiss data set, plotted in the time (top) and time-frequency (bottom) domains. Day 13, or February 23^{rd}, 2010, was one of 35 days without an avalanche.

The min, max, and median RMS events are extracted by finding the five-minute window that contains the corresponding five-second statistical RMS value. For example, Figures

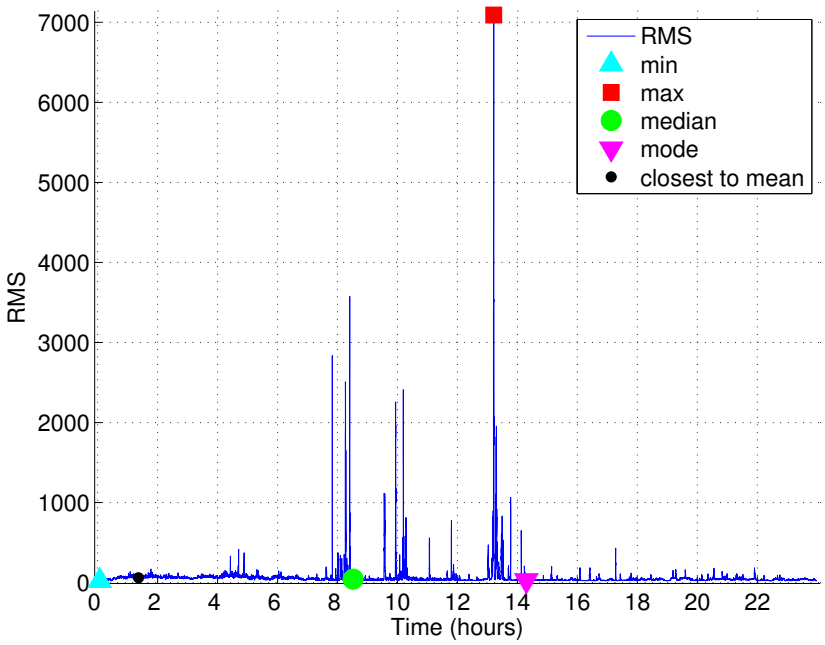


Figure 4.12: The RMS and calculated statistics for each five second frame of day 13.

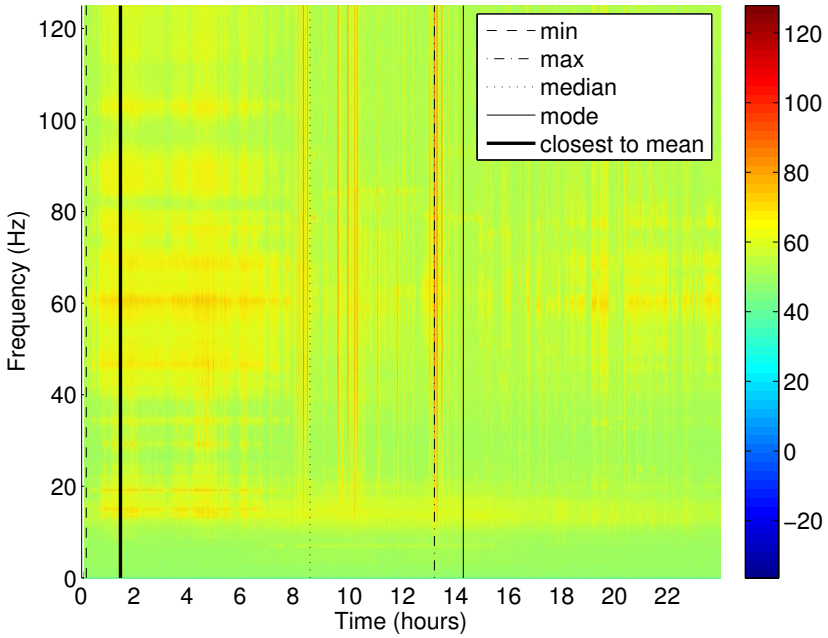


Figure 4.13: The events selected on day 13 for the five representative RMS events.

Table 4.1: The RMS events chosen for non-avalanche day 13, the starting time of the five-second frame containing the RMS event, and the actual RMS value.

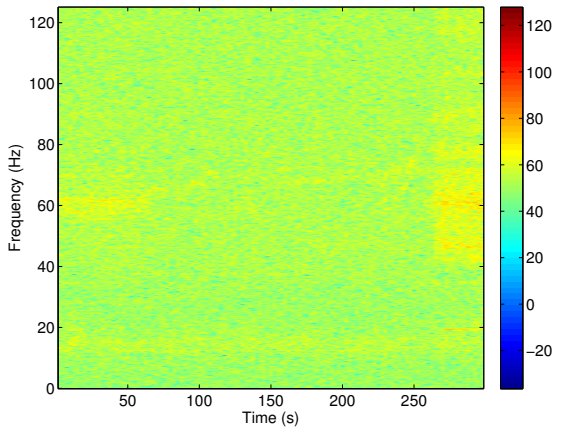
Type	Time	RMS value
Min	00:04	22.49
Max	13:13	7091
Median	08:31	42.72
"Closest to mean"	01:22	61.23
Mode	14:18	28.08

4.14(a), 4.14(b), and 4.14(c) show the min, max, and median RMS events from February 23^{rd}, 2010, respectively. In our experience, the min RMS event represents periods of prolonged "silence", the max RMS event represents a prominent non-avalanche noise event, and the median RMS event represents neither "silence" nor high energy event, i.e., some mid-range energy event such as background wind noise.

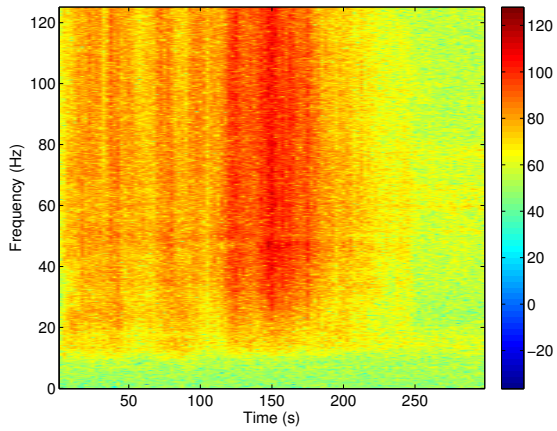
To extract the "closest to mean" RMS event, the mean RMS value for all five-second frames is calculated and the five-second frame with the smallest absolute value difference (compared to the overall mean RMS for that day) is selected. As before, the surrounding five-minutes of data (2.5 minutes before and 2.5 minutes after the "closest to mean" RMS event) are extracted and used (e.g., Figure 4.15(a)). Much like the median RMS event, the "closest to mean" RMS event represents mid-range background noise between "silence" and a high energy event.

Lastly, the mode RMS was calculated by binning all ~17,000 RMS energy values into 1000 equal sized bins (from min to max), and then selecting the most popular bin as the mode value. To determine the mode RMS event, a five-second window was selected randomly from all RMS values in the most popular bin; the five minutes of surrounding data was then included as well. The mode RMS event selected demonstrates the most common RMS energy of the day (e.g., Figure 4.15(b)).

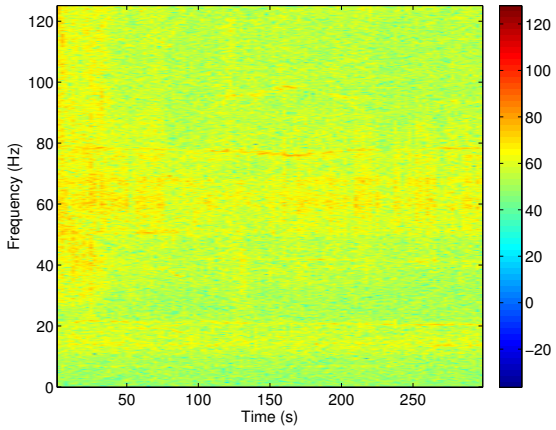
In summary, Figures 4.14 and 4.15 illustrate five selected five minute RMS events for non-avalanche day 13 of the Swiss data set. We followed the same procedure to select five



(a) min RMS event



(b) max RMS event

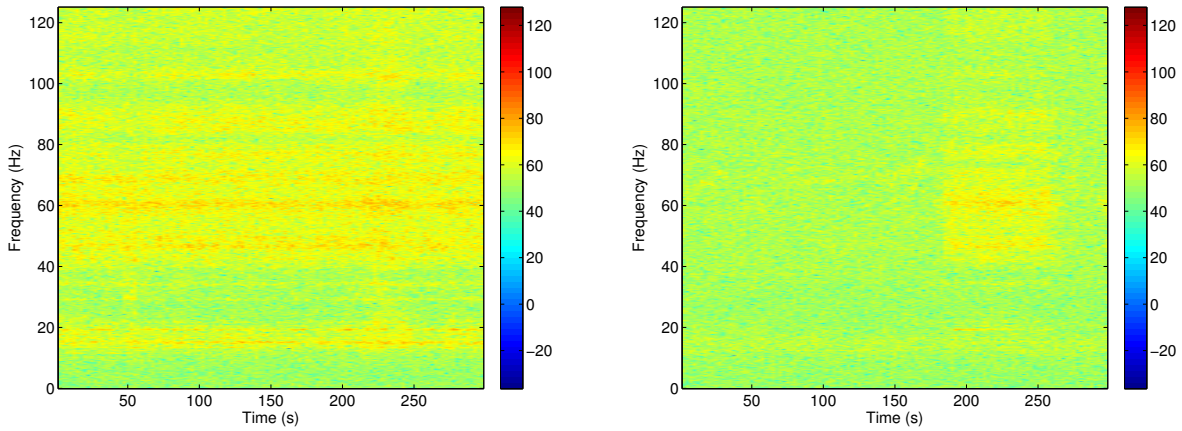


(c) median RMS event

Figure 4.14: The (a) min, (b) max, and (c) median RMS events from day 13, plotted in the time-frequency domain, showing periods of silence, a non-avalanche noise event (e.g., airplane), and mid-range noise event, respectively. The selected RMS event occurs within each five minute subset of data at time 150 seconds.

RMS events (of length five minutes each) for the other 34 non-avalanche days.

Additionally, an entire day (nearly 24 hours) of avalanche data was used in our extended analysis. This day of seismic data, i.e., April 24^{th}, 2010, contains 78 sluff avalanche events and a plethora of other noise events (e.g., Figure 4.16). We chose to simulate an entire day to help mimic how compression would perform on a complex, real-world signal that is full of small avalanche events, other noise events such as airplanes, background noises such as



(a) "closest to mean" RMS event

(b) mode RMS event

Figure 4.15: The (a) "closest to mean" and (b) mode RMS events from day 13, plotted in the time-frequency domain, showing periods of average background noise and the most common type of non-avalanche event, respectively. The selected RMS event occurs within each five minute subset of data at time 150 seconds.

wind, and silence.

To summarize, we simulated the five lossy compression algorithms on roughly 38 hours of additional seismic data. Fourteen hours of data containing 175 RMS events (35 days, five RMS events per day, five minutes per RMS event) and 24 hours of data from a full day. The algorithm parameters and evaluation metrics used were identical to those discussed in Section 4.2.

First, we evaluated the compression rates of the five simulated lossy compression algorithms executed on the six subsets of seismic data. Specifically, compression rates were calculated on each of the 175 selected min RMS events, max RMS events, median RMS events, "closest to mean" RMS events, and mode RMS events, as well as on the full day of data (Figures 4.17(a), 4.17(b), 4.17(c), 4.17(d), 4.17(e), and 4.17(f), respectively).

One key trend presented in the compression rate histograms confirms our previous results; CS and FFT, the only non-adaptive algorithms evaluated, had exclusively non-negative compression rates. Furthermore, the adaptive algorithms (i.e., KRLE, LTC, and WQTR) had

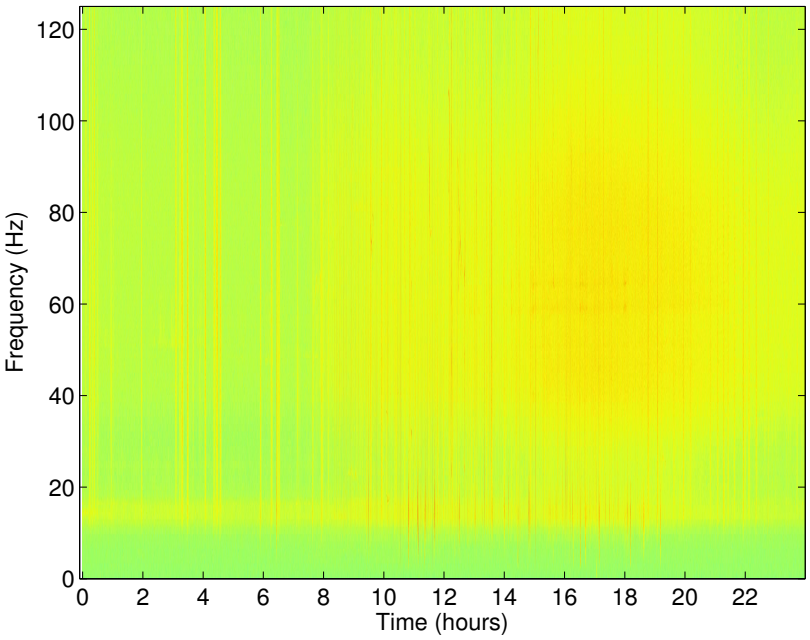
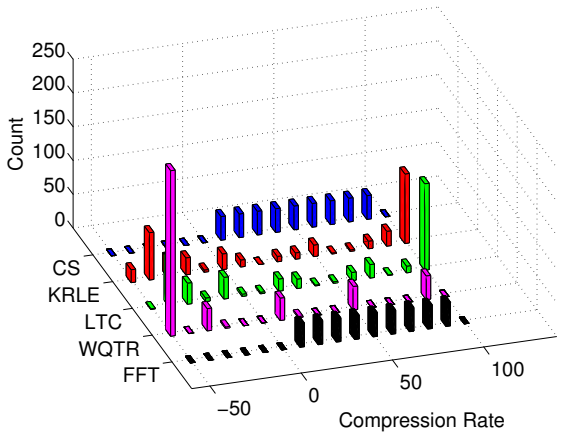
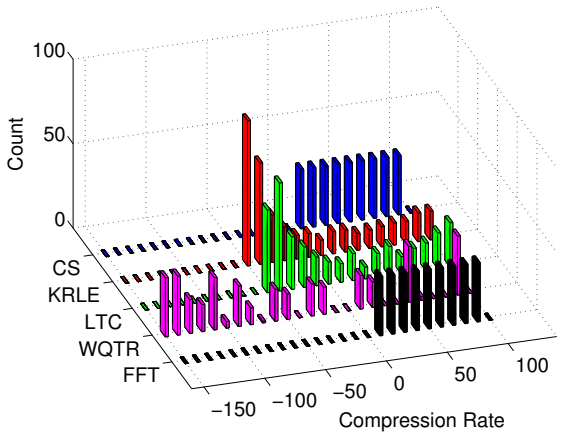


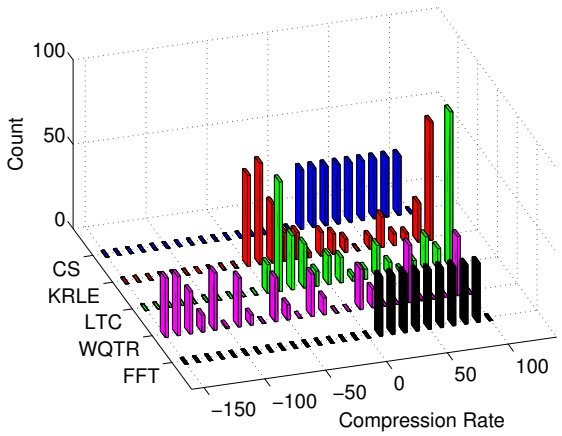
Figure 4.16: We simulated compression on an entire day (nearly 24 hours) of seismic data from April 24^{th}, 2010.



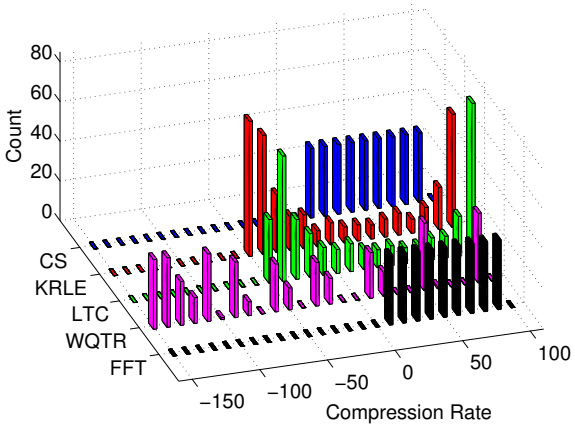
(a) min RMS



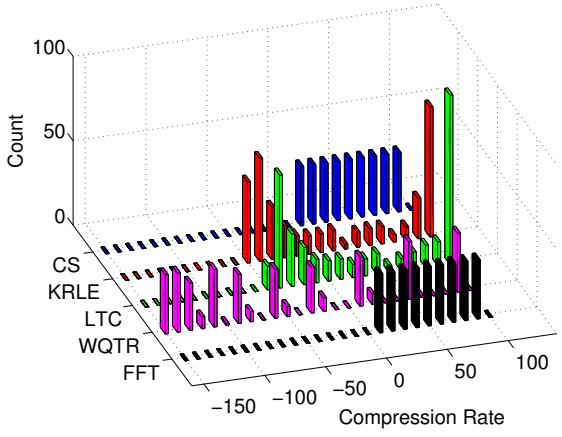
(b) max RMS



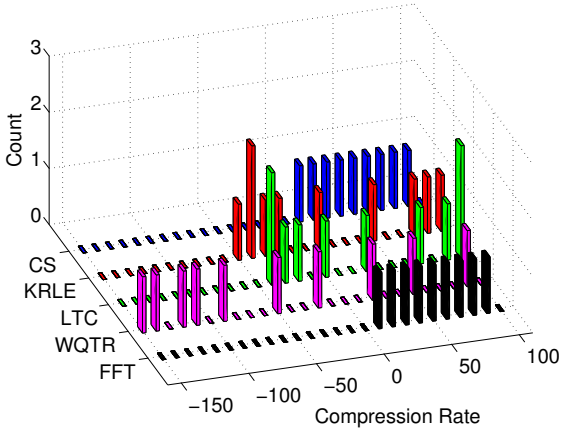
(c) median RMS



(d) "closest to mean" RMS



(e) mode RMS



(f) full day

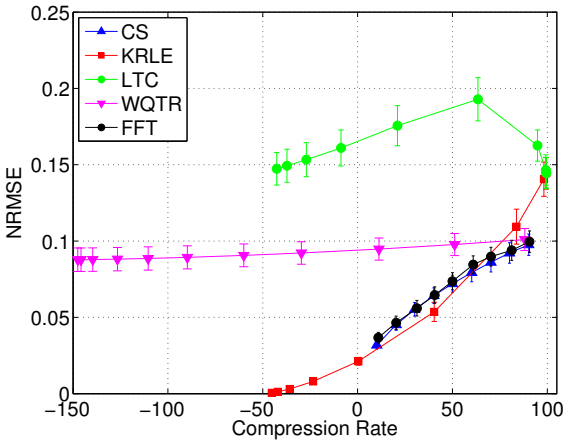
Figure 4.17: Histograms of the compression rates from simulating the algorithms on (a) min, (b) max, (c), median, (d) "closest to mean", and (e) mode RMS events, as well as on (f) a full day of data.

much more variation and possible negative compression rates. For these adaptive algorithms, the rate of compression is signal dependent. For example, KRLE performed quite poorly (in terms of compression rates) when compressing max RMS noise events (e.g., Figure 4.17(b)).

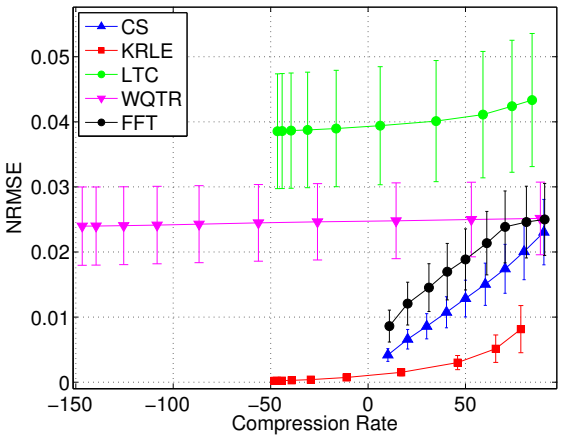
Perhaps most informative are the results depicted in Figures 4.17(e) and 4.17(f), which plot the compression rates for the mode RMS energy events and the entire day of April 24^{th}, 2010, respectively. As noted previously, the mode RMS events represent the "most common" or typical events of each day. Likewise, compressing a full day of seismic data provides a more thorough evaluation of how the algorithms will perform than our previous evaluation on only 2.75 hours of Swiss seismic data (Figure 4.2). The results demonstrate, again, the same trend described throughout this chapter; the non-adaptive algorithms had exclusively non-negative compression rates and the adaptive algorithms had negative compression rates and higher variability. These results help validate our previous analysis regarding the performance of different lossy compression algorithms.

Next we calculated the recovery error (in terms of NRMSE) for the simulated decompressed signals. Again, the methodology used to calculated NRMSE was identical to the one described in Section 4.2. The recovery error results for the min RMS events, max RMS events, median RMS events, "closest to mean" RMS events, mode RMS events, and full day are depicted in Figures 4.18(a), 4.18(b), 4.18(c), 4.18(d), 4.18(e), and 4.18(f), respectively. We note that for the full day of data, NRMSE was calculated for each five minute subset of data and then aggregated; this approach was selected to obtain a more fine-grained analysis of how each algorithm compressed and decompressed the signals.

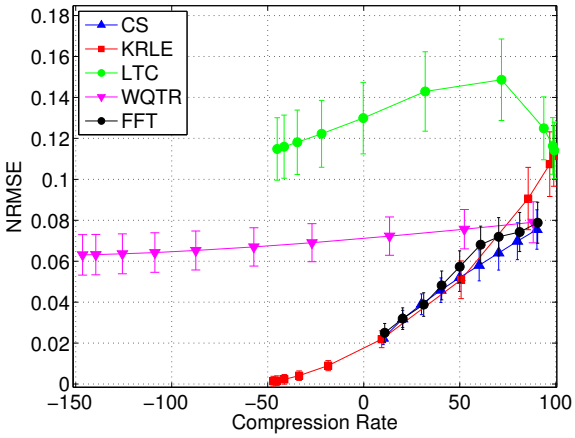
There are several trends worth noting in Figure 4.18. The first trend is the consistency of ordering between different data sets; in general, KRLE had the lowest recovered NRMSE (almost always tied with CS and FFT) while WQTR and LTC performed fourth and fifth best, respectively. This relative ordering is identical to the results presented in Section 4.2. The results confirm the viability of CS as a lossy compression algorithm for seismic data.



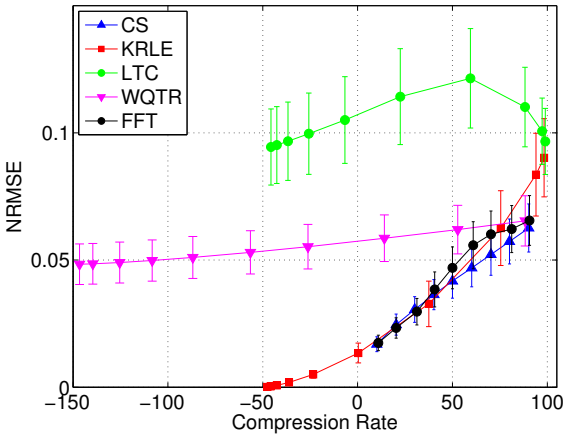
(a) min RMS



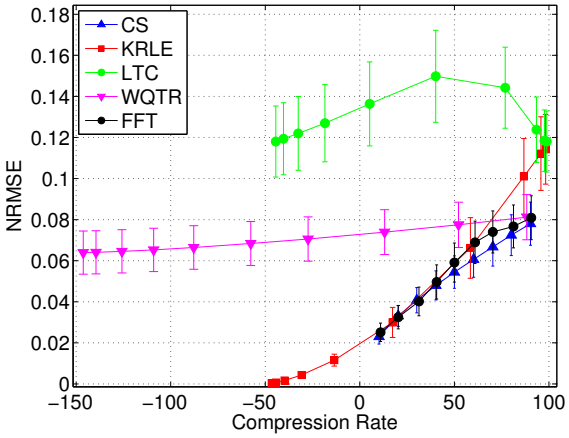
(b) max RMS



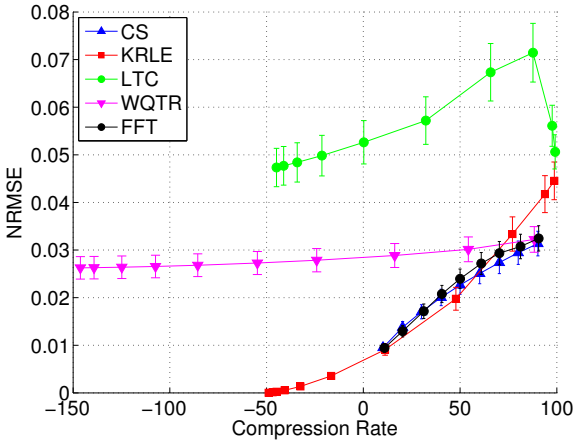
(c) median RMS



(d) "closest to mean" RMS



(e) mode RMS



(f) full day

Figure 4.18: Mean NRMSE results (with 95% confidence intervals) from simulating the algorithms on (a) min, (b) max, (c), median, (d) mean, and (e) mode RMS events, as well as on (f) a full day of data.

The second trend is the result displayed in Figures 4.18(e) and 4.18(f), which plot the NRMSE (and 95% confidence intervals) from the recovered mode RMS energy events and the entire day of avalanche data. NRMSE results from recovering the mode RMS events and full day of data show that CS is tied with both KRLE and FFT as the best performing algorithm (in terms of NRMSE). These results are important because 1) the mode RMS events are the "most common" noises found throughout the non-avalanche days and 2) simulating compression on a full day of avalanche data includes a plethora of "silence" periods and noise events typical of a complex, real-world seismic data set.

The last trend worth noting is the low recovery error of the KRLE algorithm executed on the max RMS events (i.e., Figure 4.18(b)). Recall that the max RMS events contain the highest energy non-avalanche noise events, such as a helicopter or airplane flying overhead. The maximum RMS value is indicative of large signal amplitudes and high variance. KRLE performs best on these signals (in terms of NRMSE) because the large, instantaneous "jumps" within the noise events are not compressed and, thus, do not suffer information loss. In other words, the portions of the signal with high variation (i.e., greater than K from the previous point) are encoded nearly identically to the original signal. This phenomenon explains why KRLE's compression rates are relatively low (see Figure 4.17(b)); that is, portions of the signal with high variability are not compressed very well.

In summary, the extended simulation results strengthen our argument for the viability of CS as an on-mote lossy compression algorithm for seismic data acquisition. When simulated on nearly 40 hours of additional data, CS was (in general) tied with KRLE and FFT as the best performing algorithm in terms of recovered NRMSE. Furthermore, CS and FFT were the only algorithms with exclusively non-negative compression rates, which, as discussed in Section 4.3, means that CS and FFT will *always* consume less power than full sampling. The same cannot be said for the adaptive algorithms evaluated, i.e., KRLE, LTC, and WQTR, where compression rates are signal dependent. Although both CS and FFT can guarantee power savings, as well as offer both nonnegative compression rates and acceptable

NRMSE, we note that CS outperforms FFT in terms of avalanche classification accuracy on the recovered signal (see Figure 4.4).

4.4.2 Mote Runtime Analysis

In this section, we evaluate the runtime of each algorithm while executing on the Arduino Fio wireless mote platform. In particular, our analysis examines two different aspects of the algorithms executing on mote, i.e., 1) course grain analysis of mote runtime in terms of component duty cycles and 2) fine grain analysis of mote runtime in terms of N, the size of the signal being compressed per iteration. Our analysis reveals the strengths of CS in terms of runtimes and analog to digital (AD) conversions.

We extracted component duty cycle execution, i.e., how often the mote was executing, compressing, making an AD conversion, or transmitting, by using the Fio's clock to count the microseconds of execution. Specifically, we slightly modified our compression code with wrapper functions to count the microseconds that elapsed during specific code snippets. Due to the interrupt nature of our code, several cases had to be considered for counting clock cycles.

During execution, the mote samples the ADC (and Flash memory) at 250 Hz; thus, every four milliseconds an interrupt is serviced by the mote for sampling. Additionally, when the mote is compressing the data, it is also transmitting the encoding via XBee radio, i.e., compression functions are responsible for transmitting as well. From a coarse-grain perspective, the mote is either 1) making an AD conversion when sampling (denoted herein as ADC), 2) compressing a vector of data ($Compression$), 3) transmitting the compressed encoding ($XBee$), or 4) executing other code ($Other$).

To assess execution timings accurately, we consider the following five situations that occur at some point during execution: 1) compression (cr), 2) transmissions during compression (xb), 3) AD conversions during compression (ad_{cr}), 4) AD conversions *not* during compression (ad), and 5) other. In other words, compression (cr) execution includes not only the compression algorithm, but also both radio transmissions (xb) and a portion of the

AD conversions (ad_{cr}). To calculate the total execution time of compression, transmission, and AD conversions, we used the following equations:

$$Compression = cr - (xb + ad_{cr}),$$

$$XBee = xb,$$

$$ADC = ad_{cr} + ad,$$

$$Other = exe - (Compression + ADC + xb),$$

where exe is the total execution time.

We note that compression execution time should exclude radio transmissions (xb) and the AD conversions during compression (ad_{cr}). The total execution time of the ADC includes both the ad_{cr} and ad, or analog to digital conversions that occur within and outside of compression execution, respectively. Execution of other code ($Other$) includes all clock cycles (exe) minus the time required for compression, transmission, and AD conversions; for example $Other$ includes initialization and switching buffers due to the two buffer modality described in Section 4.3.

For our component duty cycle evaluation, we executed the software presented in Section 4.3 (slightly modified, of course, for runtime component analysis). Specifically, we compressed 36 seconds of real-world avalanche data stored in Flash memory, N elements at a time; as noted in Section 4.3, we compressed $N = 256$ elements for CS, KRLE, and LTC and $N = 128$ elements for WQTR and FFT. We chose to use the most aggressive compression parameters for the adaptive algorithms; for the non-adaptive algorithms, we set the parameter to match (as closely as possible) the average compression rate for the three adaptive algorithms.

The algorithm parameters are shown in Table 4.2. In Table 4.2, DCR refers to the *desired compression rate* (between 0.0 and 1.0) for the non-adaptive algorithms CS and FFT, and Tr is the *threshold* parameter used in WQTR. For CS, DCR is inversely proportional to the

compressive sampling rate; that is, if we know DCR, we can approximate M in relation to N as follows:

$$M = floor\left(N(1.00 - DCR)\right).$$

Likewise, parameter L for FFT is set in a similar fashion (see Section 4.1.4 for details).

Table 4.2: To evaluate runtimes, we set the adaptive algorithm parameters to favor aggressive compression. We set the non-adaptive algorithm parameters such that the resulting compression rate is approximately 93.09%, the average compression rate for the three adaptive algorithms.

Algorithm	N	DCR	Parameter Value	Compression Rate
CS	256	93.09%	$M = 17$	93.36%
KRLE	256	-	$K = 512$	93.77%
LTC	256	-	$K = 512$	96.94%
WQTR	128	-	$Tr = 98\%$	88.56%
FFT	128	93.09%	$L = 4$	93.75%

Runtime execution results per component are presented in Figures 4.19 and 4.20. Specifically, Figure 4.19 plots the percentage of runtime execution for each component and Figure 4.20 depicts a "zoomed in" version to show runtime execution for AD conversions, transmissions, and compression only.

The most visible trend worth noting in Figures 4.19 and 4.20 is that CS, KRLE, and FFT spend more time doing AD conversions than executing compression. This result, best seen in Figure 4.20, speaks volumes to the efficiency of the CS, KRLE, and FFT algorithms. We note that WQTR and LTC both consumed more time executing the compression algorithms than making AD conversions. These results show the relative inefficiency of WQTR and LTC, especially compared to CS and KRLE.

Recall that CS is the only algorithm where the mote can acquire data directly in compressed form; thus, if CS is implemented to acquire data directly, the number of AD conversions will reduce. To simulate, we ignore certain AD conversions and denote this version of the algorithm, that is "compress *while* sampling", as CS*. To estimate this, we calculated the approximate execution time savings that the unnecessary AD conversions would offer.

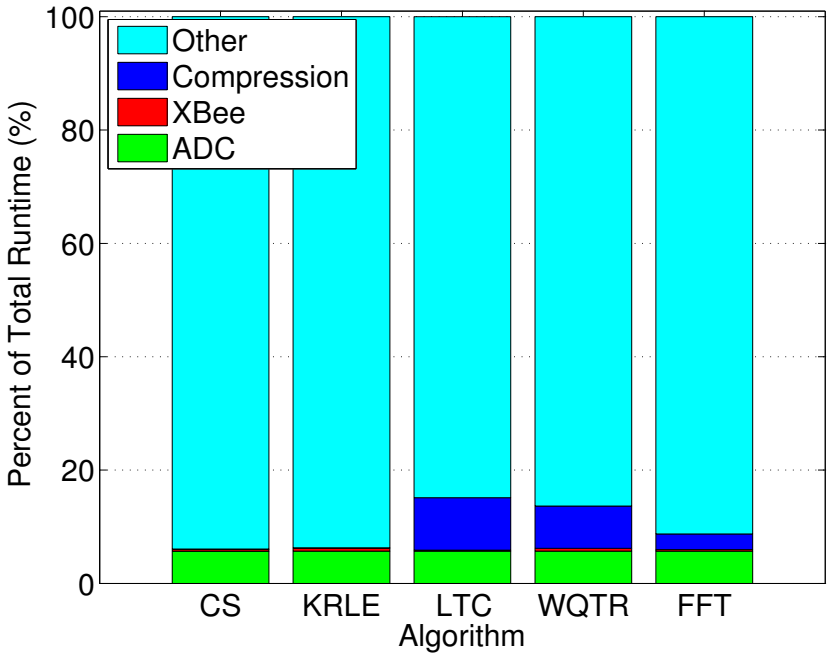


Figure 4.19: The percentage of execution runtime for each component evaluated, i.e., analog to digital conversions (ADC), radio transmissions (XBee), algorithm execution (Compression), and other (Other).

We, therefore, multiplied the total ADC runtime for CS by a factor of 6.64%, which is the percentage ratio of $M = 17$ compressive samples over $N = 256$, the total number of samples. In other words, CS* only needs to acquire $M = 17$ samples, which requires only 17 AD conversions, instead of $N = 256$ samples (and AD conversions) required for CS, KRLE, and LTC or $N = 128$ samples (and AD conversions) for WQTR and FFT. We note that $M = 17$ was chosen based on our methodology to calculate M given DCR.

Figure 4.21 compares the estimated reduced AD conversion runtimes for CS* with the other four lossy compression algorithms. As the results clearly show, CS* reduces AD conversion runtimes, thus saving both execution time and energy. Furthermore, no other algorithm known can function the way CS* does; in other words, KRLE, LTC, WQTR, and FFT require sampling the *entire signal* first.

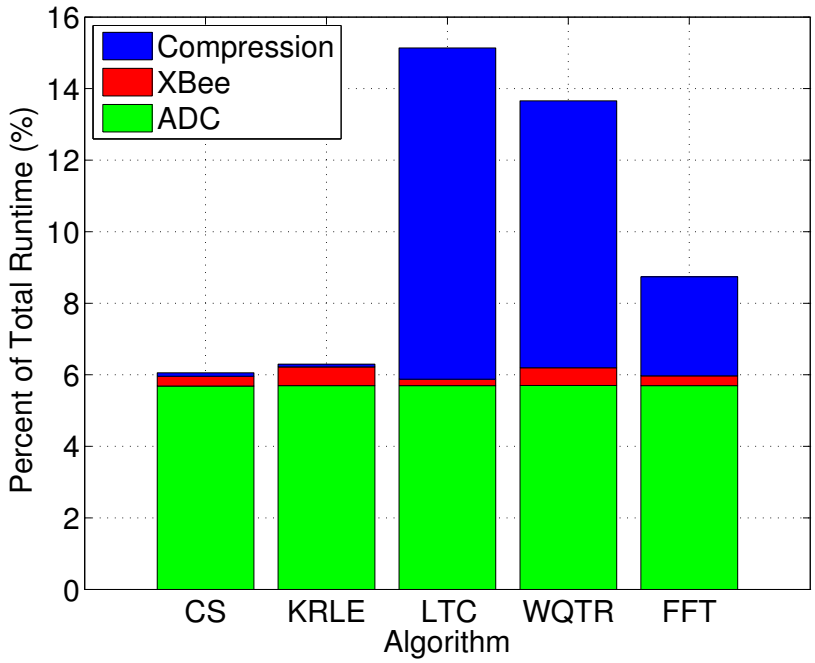


Figure 4.20: The percentage of execution runtime for only analog to digital conversions (ADC), radio transmissions (XBee), and algorithm execution (Compression).

In our second runtime evaluation, we analyzed the runtimes of compression only (not including, for example, radio runtime during the compression function). That is, we modified our compression code to eliminate ADC sampling interrupts and count the microseconds that elapsed for compression *and* XBee radio transmissions. To calculate exact compression only runtimes, we subtracted the time required for XBee transmissions from the total compression execution time. Removing XBee radio transmissions is critical to evaluating compression algorithm runtimes because the compression functions are responsible for transmitting the encodings as they are being processed. (The on-mote memory is too small to store the entire encoding before transmission). We note that the XBee transmission time should be about the same for all algorithms evaluated.

We evaluated execution times in terms of N, the size of the signal being compressed. For CS, KRLE, LTC, and WQTR, we varied N by increments of 25, from $N = 25$ to $N = 250$. For FFT, we used powers of two from $N = 16$ to $N = 256$. We did not go higher than

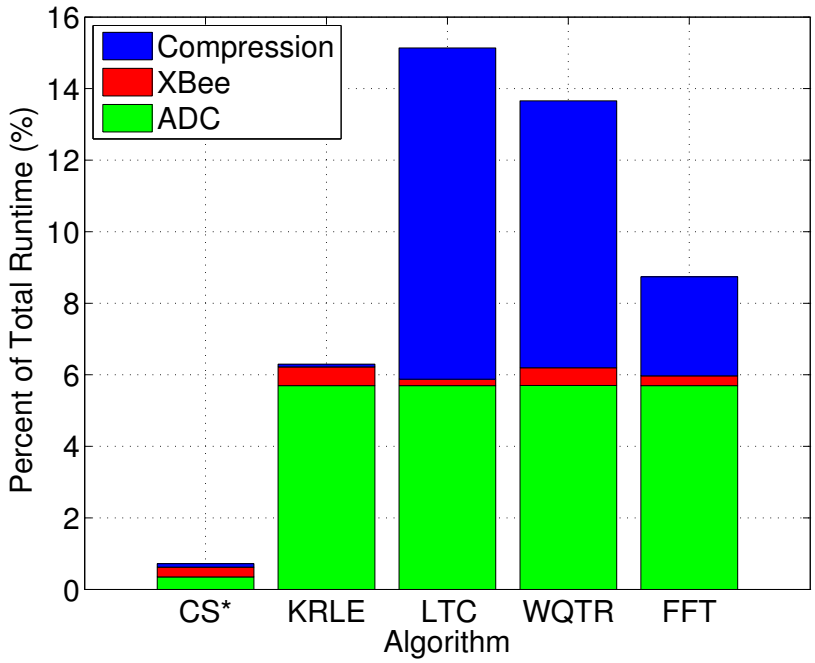


Figure 4.21: Estimated execution time of CS* when the number of AD conversions is reduced by acquiring data directly in compressed form.

$N = 256$ due to memory constraints on the mote. The test signal used in this evaluation was the same 36-second avalanche data stored in Flash memory discussed previously (see Section 4.3 for details). Runtime results presented in Figure 4.22 show the mean execution time of each algorithm as a function of N, the number of elements being compressed per iteration. In other words, we compressed the 36-second signal N elements at a time, repeating this process for each N tested.

The results in Figure 4.22 show that CS and KRLE are the most efficient algorithms evaluated (in terms of runtimes). We also note that CS's execution time can be improved; recall that CS can be implemented as CS*, where the compressed encoding is sampled directly without storing the entire signal first. CS* is advantageous in cases where AD conversions are costly in terms of time or power.

The results in Figure 4.22 also show that FFT is fairly efficient, even though the FFT was implemented in assembly (see Section 4.1.4 for details). Finally, the results show that LTC

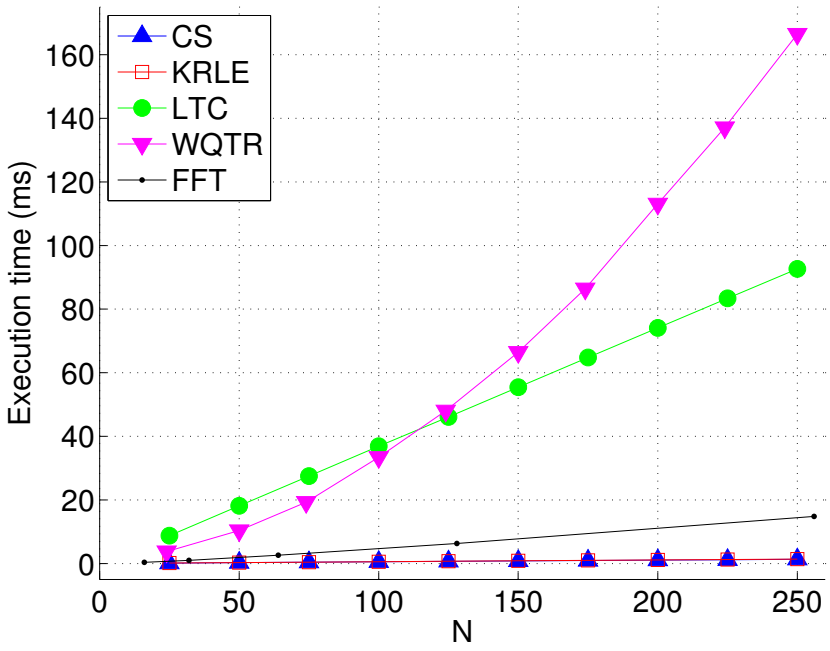


Figure 4.22: The mean runtimes of the algorithms while compressing the 36-second avalanche data N elements per iteration.

is relatively inefficient (though still linear runtime complexity) and WQTR appears to have a quadratic runtime complexity. With $N = 250$, it takes nearly 100ms to compress a signal using LTC and over 160ms to compress a signal using WQTR. These relatively slow runtimes limit the potential sampling rate on the wireless mote (in a double buffering modality); for example, if it takes less than 100ms to fill a buffer of 250 samples, then LTC will lose data.

4.5 Conclusions

In this chapter, we rigorously compared CS to four other lossy on-mote compression algorithms found in the literature. Through simulation on several real-world seismic data sets, we show that CS performs comparably to two of the four other on-mote, lossy compression algorithms (one adaptive and one non-adaptive). Additionally, we evaluated our implementation of the five lossy compression algorithms on real hardware in terms of compression rates, recovery error, power consumption, and runtime execution. Our results show that

CS, a lightweight and non-adaptive compression algorithm, performs favorably compared to KRLE, LTC, WQTR, and FFT.

We show that CS guarantees positive compression rates and reduced power consumption without sacrificing signal recovery error. Such results are promising, suggesting that CS, a non-adaptive compression algorithm with very little compression overhead, can compete with other, state of the art wireless sensor node compression algorithms. Furthermore, our classification results presented in Section 4.2.3 suggest that CS improves upon FFT in terms of the information recovered; although FFT had slightly better NRMSE results when simulated on the avalanche data, CS had less variable classification accuracies (in general). Lastly, CS can save execution time and potentially reduce power consumption by decreasing the number of AD conversions required to compress and encode the input signal.

CHAPTER 5

GEOMOTESHIELD: CUSTOM HARDWARE FOR WIRELESS GEOPHYSICS

Collecting high precision (e.g., 24-bit) seismic data for geophysical applications requires hardware components beyond the current capabilities of off-the-shelf (OTS) wireless mote platforms (e.g., TelosB [46] and Arduino Fio [47]). For example, OTS motes lack low-noise, high resolution analog to digital converters (ADCs) required for precise wireless sensing; specifically, current mote platforms have 10 or 12 bit on-chip analog to digital converters, which is not enough information for domain scientists (e.g., geophysicists) who usually demand 24-bits (or more) of precision for data acquisition.[10]

To address this insufficiency, we developed easy-to-use, custom hardware that not only provides 24-bits of analog to digital conversion, but also offers users additional features. In this chapter, we present details regarding our custom Arduino-based hardware, called the GeoMoteShield; we also compare our GeoMoteShield to a traditional wired seismograph (i.e., Geometrics Geode [48]) and another custom wireless mote our SmartGeo colleagues developed "from the ground up" (i.e., gsMote [7]). Results show that our inexpensive, easy-to-use hardware performs comparably to a much more expensive wired system and better than the more costly, "from the ground up", gsMote.

5.1 Hardware Description

In this section, we describe the design specifications for the GeoMoteShield and provide details of our implementation using OTS products. All materials, including circuit diagrams, component lists, printed circuit board (PCB) layouts, firmware, and software, will be available for open-source use.

[10] 24-bits of precision provides suitable signal sensitivity for a potentially large range of acoustic sensor values, from subtle changes on the microvolt level to large signals on the volt-level scale.

5.1.1 Specifications

In addition to providing precise analog to digital conversion, our custom hardware must be inexpensive and easy to use. In this section we list and explain the hardware specifications for our custom electronics.

- **High Precision, Low-Noise ADC:** As discussed previously, our custom hardware must provide 24-bit of signal input precision. Additionally, the ADC must have a high precision reference voltage that does not fluctuate.
- **Additional RAM:** Increasing the available RAM from 2 KB to 32 KB provides greater flexibility to the user. This additional RAM can be used to implement buffering for wireless packet collision avoidance or to perform on-mote data processing on larger chunks of the signal.
- **Persistent Removable Storage:** Current OTS platforms do not have an interface for persistent removable storage such as an SD card. Adding an optional SD card socket will provide the user with, for example, 32 GB of additional persistent storage. This increased storage space will improve the flexibility of our custom hardware; users will be able to permanently store sensor values or log files if radio communication fails.
- **Global Positioning System (GPS):** An optional GPS unit will allow users to obtain localization information as well as precise global time synchronization. As the cost of GPS microchips continues to decrease, using GPS to localize and synchronize wireless motes will increase in viability and make solving such problems as localization and synchronization much simpler.
- **Long-Range Radios:** For a real-world, outdoor deployment, sensor nodes must be capable of communicating wirelessly over hundreds of meters (or more).

The listed specifications can be addressed using an Arduino Fio mote, OTS components, and mostly open-source software libraries. In the next section we provide details of our

implementation that follows Arduino's "shield" concept.

5.1.2 Implementation

To meet the requirements discussed in Section 5.1.1 , we developed two different "shield" prototypes that can be plugged into an OTS Arduino Fio wireless mote platform. The first version of this shield, i.e., GeoMoteShield "Slim" (e.g, Figure 5.1), has a 24-bit ADC and 32 KB of external SRAM. The second version, known as GeoMoteShield "Standard" (e.g., Figure 5.2), includes a GPS unit and microSD card socket (in addition to the ADC and SRAM).

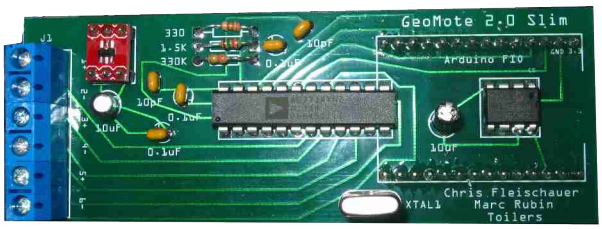


Figure 5.1: The prototype GeoMoteShield "slim" includes a 24-bit ADC and 32 KB of external SRAM. The GeoMoteShield interfaces with an OTS Arduino Fio wireless mote.

There are several advantages in creating a "shield" to interface with the existing Arduino Fio wireless platform. First, all Arduino-based platforms (including the Fio) are based on a high-level C++ based application programming interface (API) that abstracts much of the low-level procedural programming into classes and objects. This high-level abstraction provides an easy-to-use and intuitive interface to control all the components. Second, the Arduino Fio is 100% open source, both in terms of software and hardware. Thus, in the unlikely event that the Arduino Fio is no longer commercially available, we can build the wireless mote platform (with the exact same specifications) from scratch based on the open-source circuit schematic and PCB layout.

Third, the Arduino Fio has an XBee radio socket, allowing users to "plug and play" different types of radios depending on the application, from 100m range low-power radios to 10 km high power radios. Lastly, the Arduino platform has a massive online support community

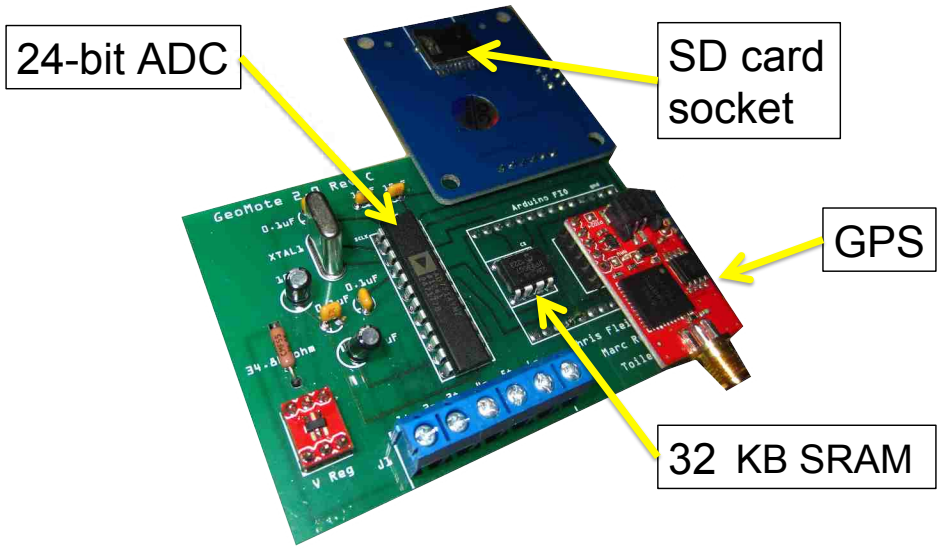


Figure 5.2: The prototype GeoMoteShield "Standard" includes a 24-bit ADC, 32 KB of external SRAM, GPS unit, and SD card socket.

that produces valuable software libraries. In our implementation we used existing software libraries for the XBee radio, GPS unit, SRAM, and SD card socket; due to the novelty of our design, we were required to develop custom firmware to control our GeoMoteShield's ADC. This is unlike the "from the ground up" gsMote, which required developing custom firmware to control not only the ADC, but also radio packetization and SRAM memory use.

5.2 Lab Tests

To validate our GeoMoteShield (which we define as the GeoMoteShield attached to an Arduino Fio with XBee radio), we conducted controlled lab tests using a WSN of nine GeoMoteShields to collect data from a voltage signal generator. Specifically, we sampled three different types of signals (i.e., sine, square, and triangle waves) with 24-bit precision at various sampling frequencies (i.e., 100, 250, 500, and 1000 Hz[11]). Thus, we analyzed data from all nine GeoMoteShields from 12 test runs (see Table 5.1). When sampling at 100 or

[11] We note that the *actual* sampling frequencies were 100, 250, 505, and 1010 Hz; the non-standard sampling rates (e.g., 1010 Hz instead of 1000 Hz) is caused by integer division of the ADC's oscillator rate. See the ADC's data sheet for specific details regarding how the sampling rate is determined [49].

250 Hz, we generated a 5 Hz test signal (approximately $\pm 0.7V$ peak to peak); when sampling at 500 or 1000 Hz, we generated a 50 Hz test signal (approximately $\pm 0.7V$ peak to peak as well). To avoid aliasing, we generated signals with frequencies well below the Nyquist rate.

Table 5.1: In our lab tests, we used a WSN of nine GeoMoteShields to simultaneously record three different types of waveforms produced by a signal generator. We varied GeoMoteShield sampling rates and adjusted the waveform's frequency to avoid aliasing. In sum, we analyzed 12 waveform signals recorded by nine GeoMoteShields for a total of 108 traces.

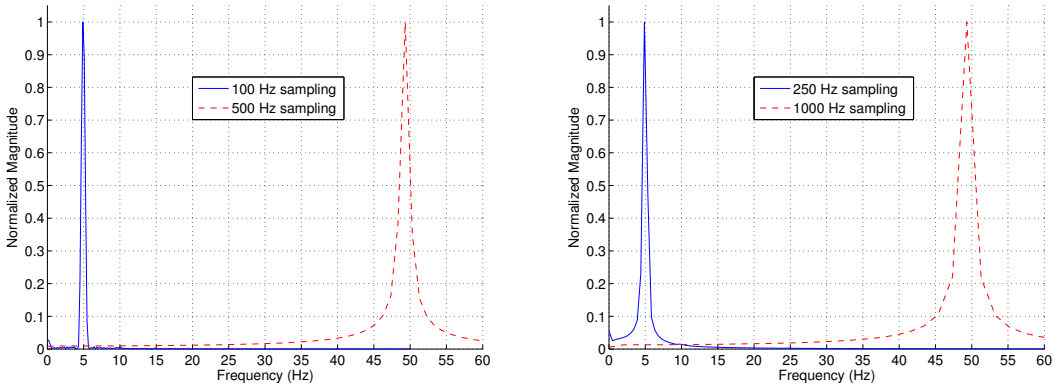
GeoMoteShield Sampling Rate (Hz)	Waveform	Waveform Frequency (Hz)
100	sine, square, triangle	5
250	sine, square, triangle	5
500	sine, square, triangle	50
1000	sine, square, triangle	50

We confirmed the relative accuracy of the GeoMoteShield ADC's clock by qualitatively analyzing the acquired sine wave data in the frequency domain. More precisely, for each of the four sampling rates evaluated, we "stacked" the nine acquired signals by summing them together and, after subtracting the mean to remove the DC component (i.e., also known as the bias or offset voltage), computed a 512-bin FFT. Visual inspection of the acquired signals in the frequency domain (Figures 5.3(a) and 5.3(b)) show that the acquired signals were very close to the target frequencies (i.e., 5 Hz and 50 Hz).

For each combination of input signal and sampling frequency, the acquired samples were then evaluated in terms of signal precision and time synchronization error (between GeoMoteShields). We detail our results in the next two sections.

5.2.1 Precision

We initially compared the similarities and differences of the data recorded by the nine GeoMoteShields. This evaluation was used to determine the precision between multiple GeoMoteShields; in essence we seek to answer the question: "are the signals acquired by all GeoMoteShields basically the same?".



(a) (b)

Figure 5.3: The normalized frequency magnitudes of a 512-bin FFT for the two sine waves (i.e., 5 Hz and 50 Hz) acquired using (a) 100 and 500 Hz sampling rates and (b) 250 and 1000 Hz sampling rates.

To answer this question, we calculated the normalized root mean square error (NRMSE) between all pairs of signals collected for each combination of signal and sampling rate. See Section 3.3.1 for details regarding how NRMSE is calculated between an original (ground-truth) signal and an experimental signal. For example, Figure 5.4 shows 5 Hz sine, square, and triangle waves acquired from all nine motes at 100 Hz sampling rate and 24-bit precision. Figure 5.5 then plots the "stacked" signals, such that all nine signals are placed on top of each other. Visual inspection of Figure 5.5 indicates that the GeoMoteShield performs quite well. That is, data acquired from nine GeoMoteShield WSN shows high signal precision and time synchronization; the signals are all very similar and time aligned.

For each run, the NRMSE was calculated for each combination of signal and sampling rate; for example, the following pairs of GeoMoteShield acquired samples were analyzed: $\{(1,2),(1,3),...,(2,3),(2,4),...,(7,9),(8,9)\}$, where 1 represents data collected by GeoMoteShield1, 2 represents data collected by GeoMoteShield2, etc., (see Table 5.2). Figure 5.6 plots a statistical box plot summary (median, 25% and 75% quartiles, 2.7σ, and outliers) of the NRMSE for each signal type.

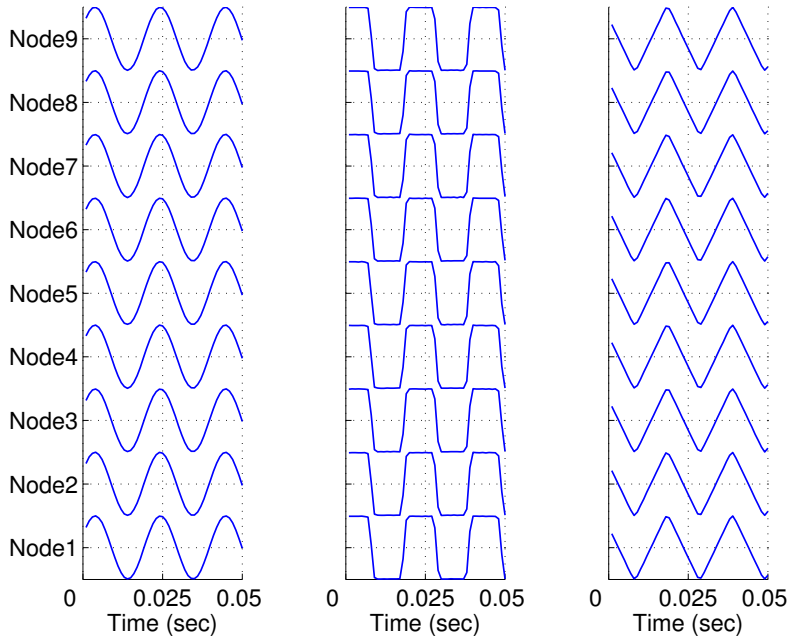


Figure 5.4: Examples of signal data acquired simultaneously using nine GeoMoteShields. In this figure, the 5 Hz sine, square, and triangle waves were sampled at 100 Hz with 24-bit precision.

Table 5.2: Within each test run, all non-repeating combinations of acquired signal pairs were analyzed for precision and time synchronization. In this table, '1' denotes the signal acquired by GeoMoteShield1, '2' denotes the signal acquired by GeoMoteShield2, etc.

Signal	Paired with Signal
1	2, 3, 4, 5, 6, 7, 8, 9
2	3, 4, 5, 6, 7, 8, 9
3	4, 5, 6, 7, 8, 9
4	5, 6, 7, 8, 9
5	6, 7, 8, 9
6	7, 8, 9
7	8, 9
8	9

The NRMSE results show that the GeoMoteShields are precise. The median NRMSEs for all three types of signals are below 0.005 (or 0.5% error, relative to the signal range).

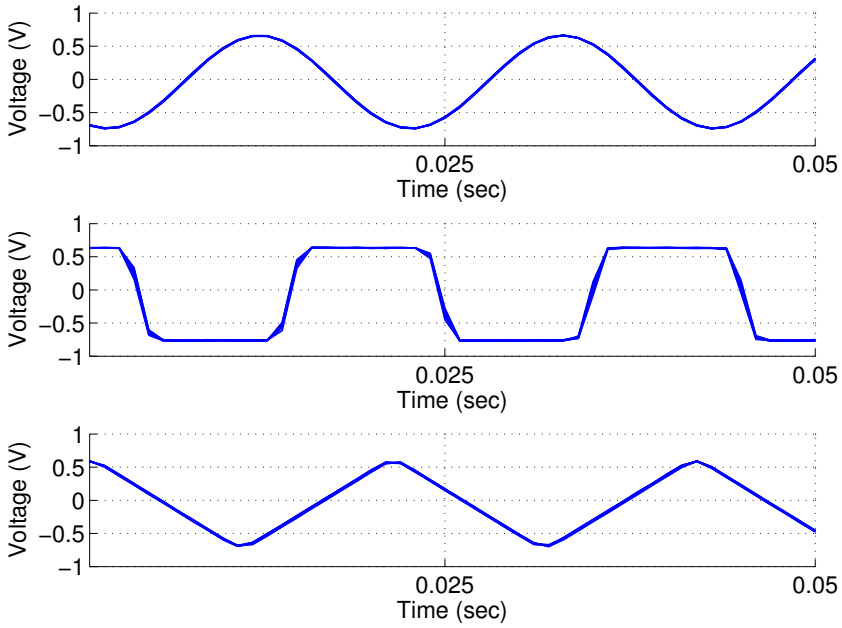


Figure 5.5: "Stacked" lab test data acquired from nine GeoMoteShields.

5.2.2 Time Synchronization

Time synchronization of the samples collected is critical for geophysical modeling, inversion, and visualization. It is our understanding that, in geophysics, the time lag between multiple signals is very important for understanding the data; time lags often correspond to arrival time differences of the acoustic wave(s) being measured. Thus, a sensing system must be capable of very accurate time synchronization such that the offsets between acquired signals are caused by geophysical phenomena and not time synchronization errors. We estimated time synchronization errors of our custom GeoMoteShields by calculating the cross-correlation time lag between all unique pairs of the acquired signals.

Cross-correlation is a common procedure used in signal processing to estimate the time delay between two similar signals [40]. In essence, computing the cross-correlation time lag indicates the amount of time shift required for one signal to have maximum correlation with another signal. This time shift, also known as the lag, offset, and/or delay, can be used to

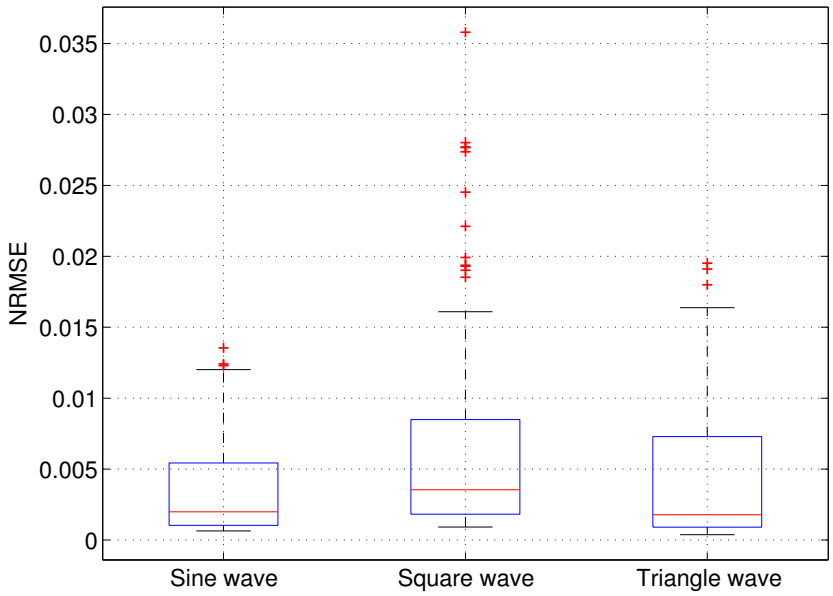


Figure 5.6: Box plots showing a statistical summary of the calculated NRMSE. The red line is the median, the blue box represents the 25^{th} and 75^{th} percentiles, the "whiskers" correspond to 2.7σ (where σ is the standard deviation), and the red points are outliers.

estimate how well the GeoMoteShields are synchronized in time; two perfectly aligned (and time synchronized) signals would have zero time lag offset. Our aggregated results, shown in Figure 5.7, indicate that we had perfect time synchronization, i.e., *zero* offset between *all* pairs of the nine signals acquired during each run (see Table 5.2 for pairing details).

Given the success of our Arduino-based GeoMoteShield (in terms of NRMSE measuring signal precision and cross correlation time lag measuring time synchronization), we proceeded to further test the GeoMoteShield in more realistic tests (e.g., using geophone sensors). We also compare our wireless, Arduino-based GeoMoteShield to a second wireless geophysical mote and a wired geophysical sensing system. In the next section we summarize and analyze results comparing two wireless sensor networks and one wired seismic acquisition system.

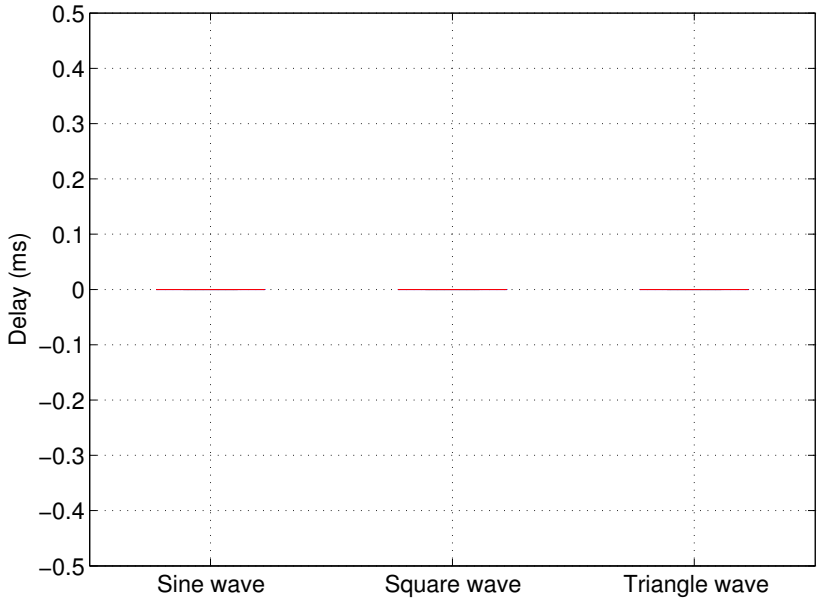


Figure 5.7: Statistical summary of all cross-correlation time lags between lab signals acquired using nine GeoMoteShields.

5.3 Field Test 1: Walk Away

In this section, we summarize results from a field test comparing a traditional wired acquisition system (i.e., Geometrics Geode Seismograph) to our custom, Arduino-based GeoMoteShield and a "from the ground up" wireless geophysical mote (i.e., gsMote) developed by our SmartGeo colleagues [7]. In particular, we recorded seismic data (using all three systems) during a "walk-away" seismic field test and then compare the wired versus wireless data in terms of signal quality. Field testing was led by a geophysics expert and Ph.D. student at Colorado School of Mines.

5.3.1 Experimental Setup

The experimental design, known as a "walk away" test in geophysics, consists of placing many geophone sensors adjacent to one another in a cluster with minimal spacing. As data is being acquired for each test, the experimenter uses a sledgehammer to pound a metal

plate on the ground three times and the test is then repeated at various distances from the cluster. The purpose of the "walk away" test is to evaluate and calibrate an acquisition system for data consistency and time synchronization accuracy; the sledgehammer impact waveform *should* be nearly identical in all sensors used.

For our experiments, we used three different systems, a Geometrics Geode *wired* system, a WSN of SmartGeo's custom gsMotes, and a WSN of our GeoMoteShields. For each system, we acquired data simultaneously from nine geophone sensors placed in a tight cluster for the "walk-away" test (see Figure 5.8). The wired data was collected via nine channels; for simplicity in comparison, we refer to these channels as nodes in our analysis. Each geophone was sampled at 1000 Hz sampling rate with 24-bit precision, the GeoMoteShield's gain was set to 32, and gain for both the gsMote and wired systems was set to one (no gain). We selected a mid-range gain for the GeoMoteShield due to preliminary field testing and geophysicist expert opinion; a gain of 32 (out of 1, 2, 4, 8, 16, 32, 64, and 128) produced the best data, according to the geophysicist leading the field tests. We note that, unlike our GeoMoteShields, the gsMotes did not have have the capability to switch gain settings "live" during field use; thus, a conservative gain setting for the gsMote was selected by SmartGeo colleagues prior to field testing.

The sledge hammer impacts were delivered at 10 and 20 feet from the geophone cluster. The results presented are from tests four and seven of seven (i.e., sledgehammer impacts 20 feet from the target). We note that runs four and seven were selected because they were the only tests with full and synchronized data from *all* three systems. See Table 5.3 for details on the seven "walk-away" tests. In the WSNs, each geophone sensor was interfaced with a single wireless mote; thus, each WSN consisted of nine sensing nodes and a base station to receive and store the data. In total, we acquired data from 27 identical 40 Hz geophones during seven tests.

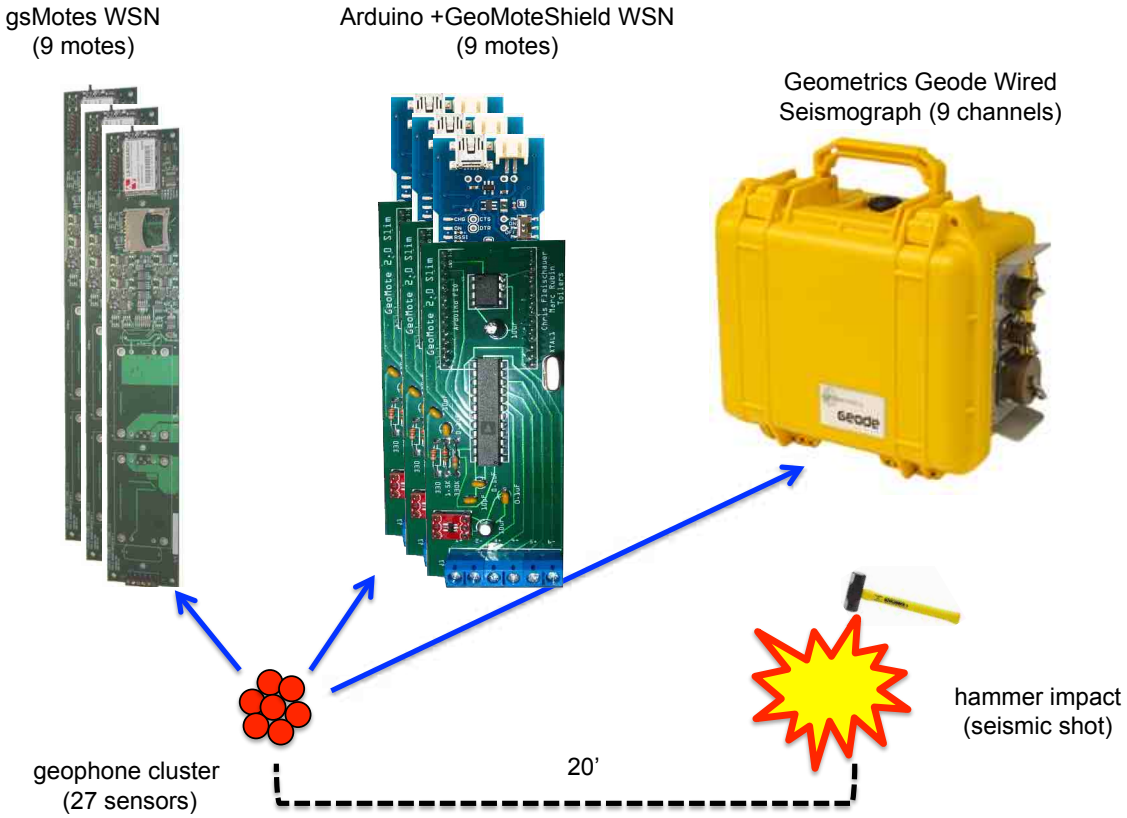


Figure 5.8: We performed seismic "walk-away" field tests to compare three sensor systems: a WSN of "from the ground up" wireless gsMotes, a WSN of Arduino-based GeoMoteShields, and a Geometrics Geode wired seismograph.

Table 5.3: Summary of the seven "walk-away" tests including the distance of the three sledgehammer impacts as well as comments regarding data reliability.

Test Number	Impact Distance (feet)	Comments
1	10	No data from gsMote WSN.
2	10	No data from gsMote WSN.
3	10	Missing data from gsMote #4.
4	20	OK.
5	20	Missing data from gsMotes #2 and #9.
6	20	Missing data from GeoMoteShield #7.
7	20	OK.

5.3.2 Signal Preprocessing

In this section we present details for how we prepared the acquired seismic signals for comparison. During field test data acquisition, the three systems were not perfectly time aligned; in other words, "time zero" of the three systems was not exactly the same. The imprecise time alignment between the three systems was due to the systems not being controlled by a single computer or user. To account for this imprecision in acquisition start times, we pinpointed the three seismic "events" (or sledge hammer impacts) recorded during each test run. We then time aligned the seismic events using a cross-correlation lag approach common in signal processing [40].

More specifically, we analyzed the signal envelope of each node's data and picked the three most energetic points (which correspond to the hammer impacts, e.g., Figure 5.9). The signal envelope is generated by interpolating between local absolute value maxima of consecutive (non-overlapping) signal subsets of size 16 samples. In our experience, this default parameter value worked well for our intentions of locating the start of each seismic event; thus, we did not explore other parameter values. After finding the three peaks from the three sledgehammer impacts in the signal envelope, we extracted 100 ms of seismic data with the point of each peak occurring at the halfway point, i.e., at 50 ms. All preprocessing was completed with Matlab and the open-source MIRToolbox [20] for time series signal processing.

Next, as is common in geophysical analysis, we normalized each seismic signal to have a common scale of zero mean and unit variance, i.e.,

$$x_n = \frac{x - mean(x)}{std(x)},$$

where x is the original data and x_n is the normalized data. We then time aligned the wireless signals to the wired data by calculating the cross-correlation time lags between the respective signal peaks, finding the offset with maximum correlation, and applying the appropriate delay to the wireless signal. Figure 5.10 illustrates the signal processing steps

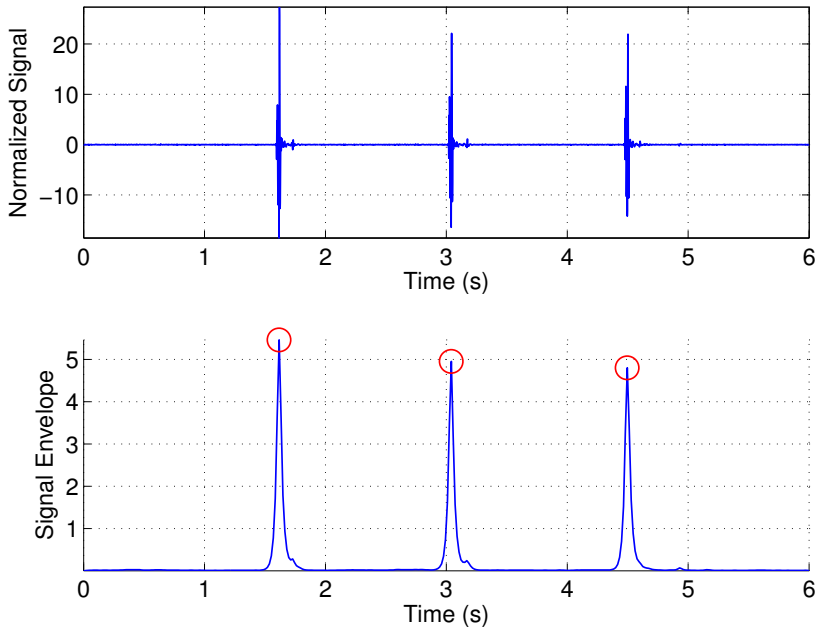


Figure 5.9: The raw seismic data (top) and the estimated signal envelope with three peak events selected (bottom).

for the cross-correlation and time alignment of the GeoMoteShield data. We note that the signals were aligned appropriately; in other words, wired data from node one was time aligned with data from both GeoMoteShield one and gsMote one, etc.

Finally, Figure 5.11 depicts the time-aligned and normalized seismic events between nodes and systems. Now that the signals of interest are normalized and time aligned, we analyze signal accuracy *between* the wired and wireless systems as well as signal precision and time synchronization *within* each system. The next three sections provide our results.

5.3.3 Accuracy

To determine the accuracy of the GeoMoteShield and gsMote, we calculated the normalized root mean square error (NRMSE) between the seismic events acquired with the wired and wireless systems. For each test run evaluated, we calculated the NRMSE between the seismic events recorded by wired node one and GeoMoteshield one, wired node two and

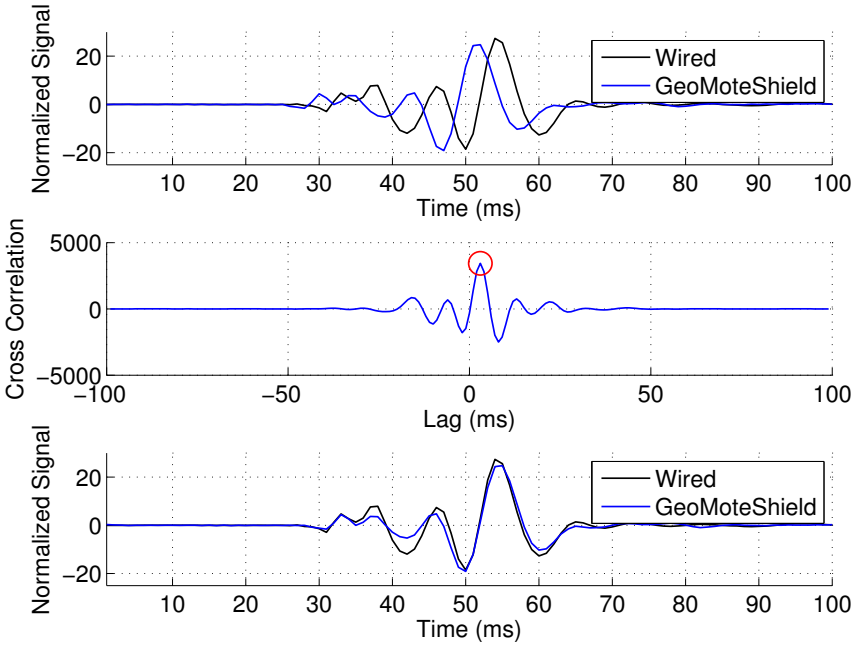


Figure 5.10: The Wired and GeoMoteShield data from Node1 were not time aligned (top). Cross-correlation was used to estimate the time lag between signals (middle). The Wired and GeoMoteShield signals after the appropriate offset was applied to the wireless data (bottom).

GeoMoteShield two, etc. Likewise, we compared data from wired node one and gsMote one, wired node two with gsMote two, etc. Accuracy corresponds to how close the sensor system's data is to physical reality; in our case, we assume that the wired data is "ground truth". Figure 5.12 summarizes the NRMSE results from the wired versus wireless signal comparisons; box plots are used to plot the median, 25% and 75% quartiles, 2.7σ, and outliers.

The accuracy results presented in Figure 5.12 show that our GeoMoteShield performs comparably to the "from the ground up" custom gsMote. In other words, an Arduino-based system is just as accurate as (if not more accurate than) a more expensive, fully custom wireless solution. Additionally, the GeoMoteShield is capable of recording fairly accurate signals close to actual "ground truth" (i.e., wired data).

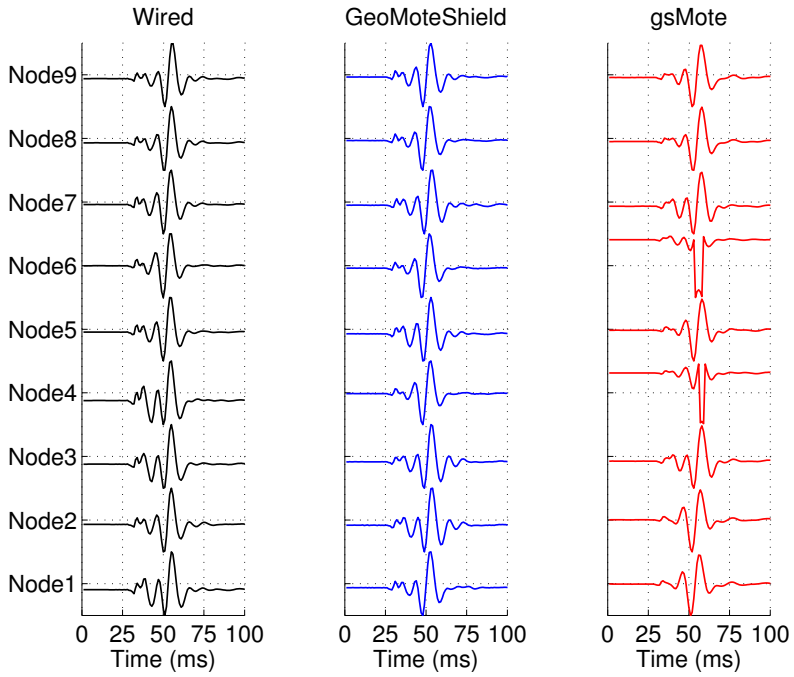


Figure 5.11: One of the three sledgehammer impacts normalized and time-aligned seismic events recorded by all three systems during the "walk-away" field test. The Wired system is a Geometrics Geode seismograph, the GeoMoteShield is our custom, Arduino-based shield, and the gsMote is a standalone, "from the ground up" mote developed by SmartGeo colleagues.

5.3.4 Precision

We next calculate the precision of the GeoMoteShield, gsMote, and wired systems. Precision represents the reproducibility of a sensor system; a system with perfect precision would acquire the same signal every time (though this data may not be accurate, per se). Precision is critical to geophysical data interpretation; differences between multiple time synchronized signals *should* be caused by differing sensor activations, not imprecision between individual nodes.

To estimate precision, we evaluated signal differences (in terms of NRMSE) between all unique combinations of pairs "within" each sensor system per test run (see Table 5.2 for pairing combinations within each system). Figure 5.13 depicts a statistical summary of the combined NRMSE results. We reiterate that all signals were normalized to have mean zero

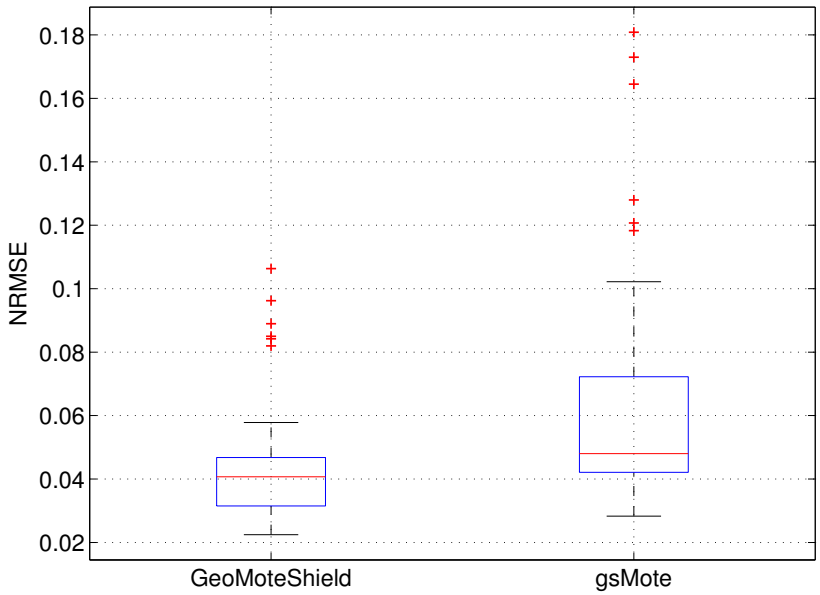


Figure 5.12: Box plots showing a statistical summary of accuracy (in terms of NRMSE) comparing the seismic event signals between the two wireless systems and the wired system.

and unit variance before evaluation.

The results in Figure 5.13 show that our GeoMoteShield offers similar precision to the wired system. That is, a WSN of our GeoMoteShields will record data that is very similar and reproducible between individual sensor nodes. Our results show that, while there is *some* imprecision between sensor readings for the wired system (most likely caused by geophone output differences), the precision rates of our GeoMoteShields are similar.

5.3.5 Time Synchronization

Figure 5.14 plots the nine stacked seismic signals acquired with all three systems from test run four. The figure shows that, even in the wired case, there are very subtle time offsets between the signals obtained. Understanding the mechanisms responsible for such signal offsets fall beyond the scope of this dissertation and into the domain of geophysics; it is highly unlikely that the time lag offsets present in the wired data is caused by time

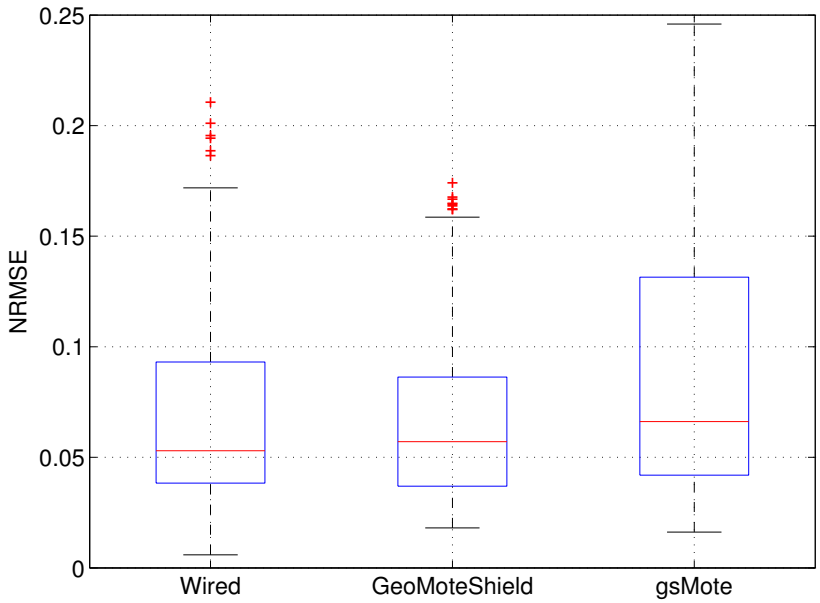


Figure 5.13: Box plots showing a statistical summary of precision (in terms of NRMSE) comparing the seismic events "within" each sensor system, i.e., between each unique combination of node pairs.

synchronization errors.

As in the previous section and denoted in Table 5.2, we evaluated time synchronization by comparing the time lags between all unique combinations of signal pairs "within" the three sensor systems. We estimated time lag by calculating the cross-correlation between each signal pair and finding the offset where the maximum correlation between signals occurred. Using this method, we calculated the time lags of the data from the wired system and compared it to the signals acquired with the two wireless systems. Figure 5.15 plots a statistical summary of all cross-correlation offsets calculated within each system.

Our results show that, in terms of time synchronization, our GeoMoteShield is just as accurate as the wired system and the "from the ground up" gsMote. It is important to note that WSN time synchronization was implemented in a "naive" way, using the base station to control when motes start and stop collecting data. Based on these results, we

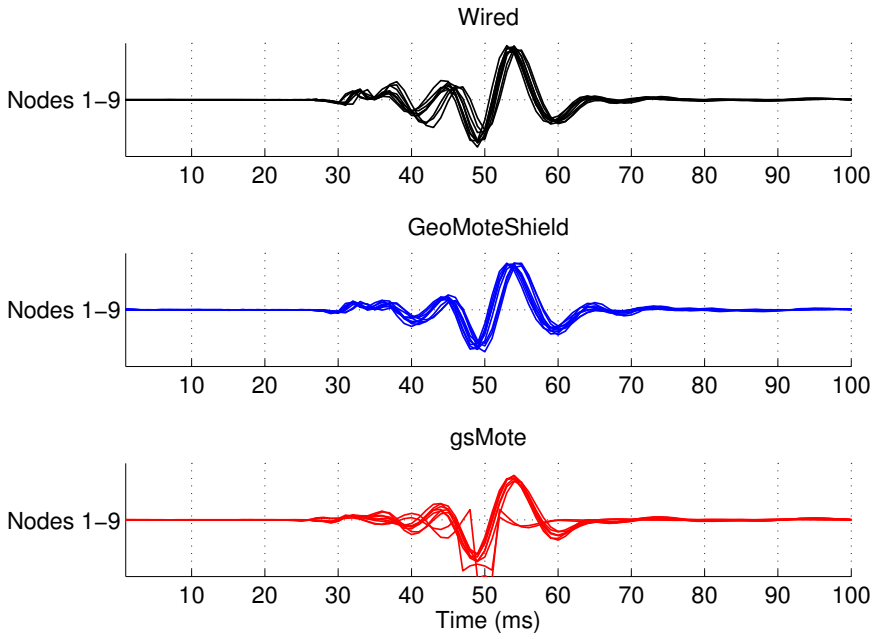


Figure 5.14: "Stacked" signals from all nine nodes illustrating a sledgehammer impact event.

can safely assume that delays from radio propagation and reception are negligible, i.e., our GeoMoteShield had virtually the same time lag offsets between acquired signals as the wired system. We note that, for a long-term WSN field deployment with continuous sampling, more intelligent time synchronization must be implemented. Additionally, a temperature compensated real-time clock is needed to account for wireless mote clock drift.

5.4 Field Test 2: Refraction Survey

The second field test consisted of a typical seismic refraction survey, with nine geophones placed in a linear array at 10 foot spacing (80 feet overall). During data collection, a sledgehammer was used to deliver a seismic "shot" directly adjacent to one of the geophone sensors while the other sensors "received" the seismic signal (see Figure 5.16). We collected data at 1000 Hz sample rate with 24-bit precision using a wired Geometrics Geode seismograph and a WSN of our GeoMoteShields. We omitted the gsMote WSN from our second field test; as shown in Section 5.3, our Arduino-based system performs comparably to a traditional,

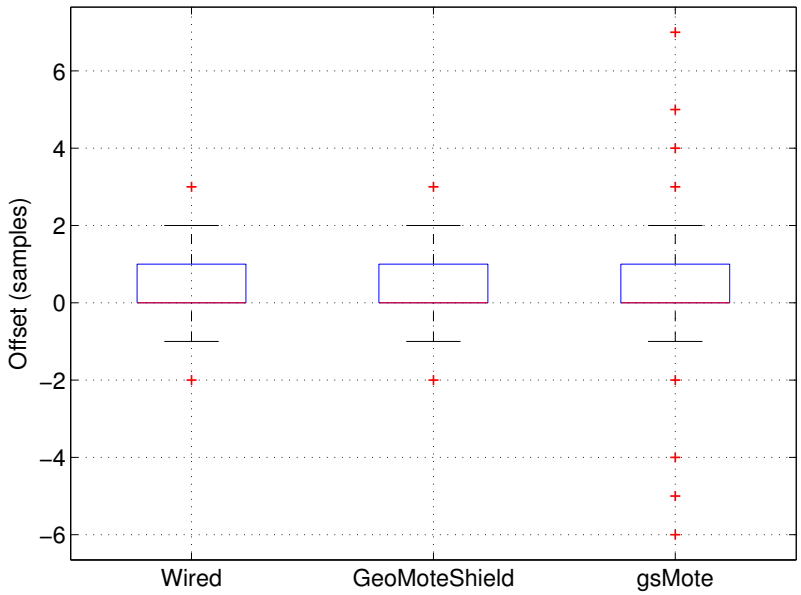


Figure 5.15: Box plots showing a statistical summary of the time lag offsets between acquired signals "within" each sensor system tested. Note that zero offset would indicate perfect time alignment.

wired system and better than SmartGeo's fully custom, "from the ground up" gsMote. We also note that the GeoMoteShield is much cheaper and significantly easier to use than the gsMote.

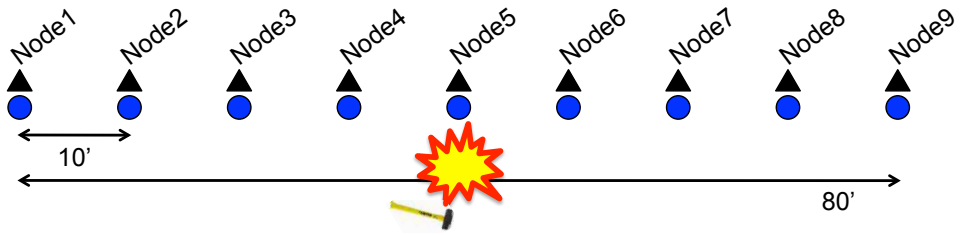


Figure 5.16: In our second field test, nine geophones were spaced 10 feet apart and a seismic "shot" was delivered adjacent to each geophone location. Data was recorded simultaneously by both the wired and wireless systems (indicated by the black triangles and blue circles, respectively).

5.4.1 Experimental Setup

As noted, our experiments consisted of recording wired and wireless data simultaneously during a typical refraction survey; specifically, seismic "shots" were delivered adjacent to each of the nine geophone locations. Snapshot data was recorded once per seismic "shot" location, giving a total of nine full and synchronized sets (81 total traces) of seismic data to analyze per system. Figure 5.17 depicts a typical seismic "shot" at node five, as received by the wired seismograph and WSN of GeoMoteShields. The pattern of the seismic wavefront can be seen in Figure 5.17; geophones further from the "shot" receive the waveform slightly later in time.

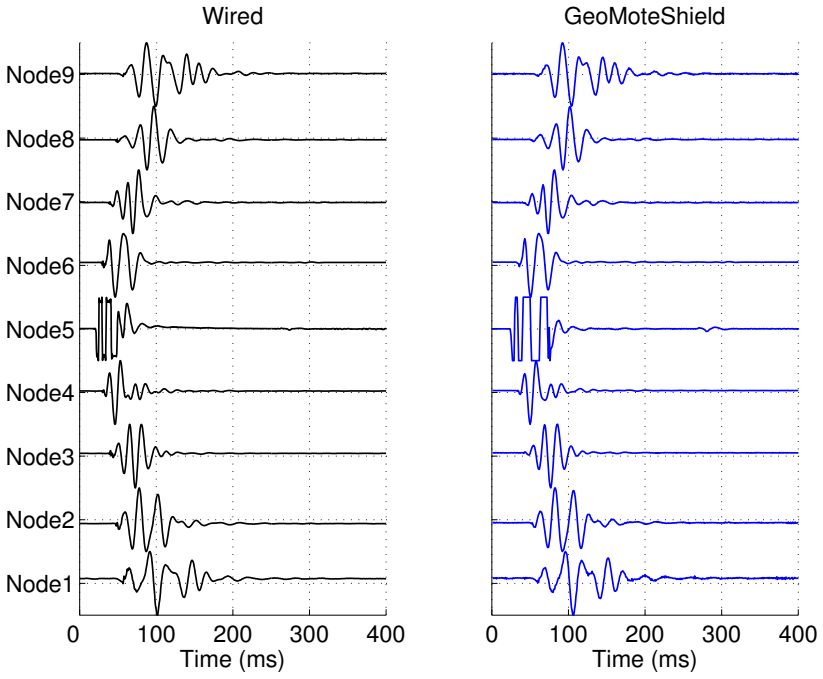


Figure 5.17: Seismic data recorded by all nine nodes of the wired and wireless systems when the seismic "shot" occurred at node five.

5.4.2 Arrival Times

When geophysicists analyze seismic data, the first step is often finding the arrival times of the seismic pressure wave at each geophone. Geophysicists use these arrival times (along

with wavelet information) to perform inversions and visualize the earth's subsurface. For our analysis, the seismic "shot" arrivals were picked manually. Figure 5.18 depicts a scatterplot of the estimated[12] arrival times given the distance between "shot" and "receiver" pairs. We note that there are only two data points at 80 feet (per system) because there were only two instances during testing where the seismic "shot" was 80 feet from the receiver (i.e., "shot" at geophone one, receive at geophone nine and "shot" at geophone nine, receive at geophone one).

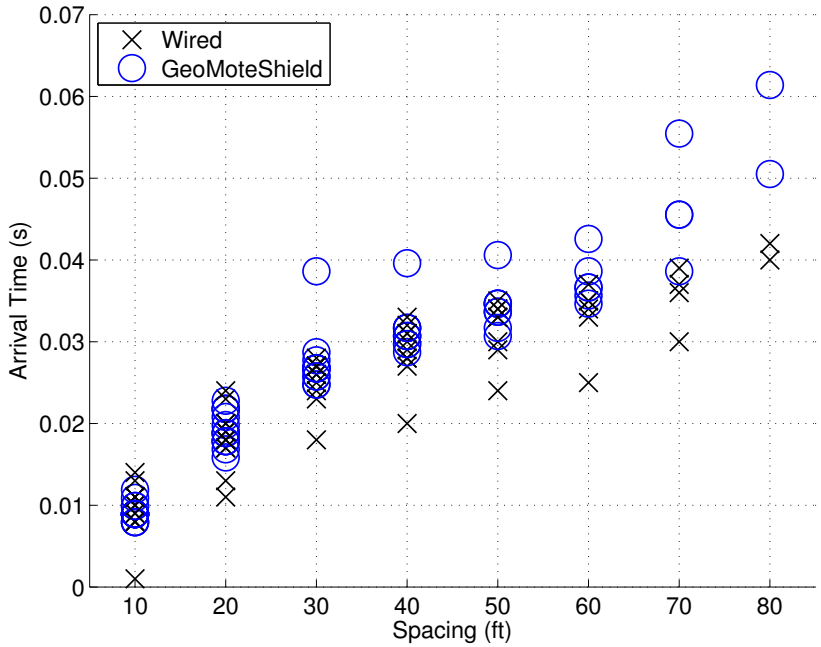


Figure 5.18: The estimated arrival times of the seismic energy for each shot-receiver pair (by distance). Note that some arrival times overlap.

In Figure 5.18, note the "outlier" arrival times for the GeoMoteShield and Wired systems between 30 and 70 feet (inclusive). These outliers show instances when the arrival times are higher for the GeoMoteShield and lower for the Wired systems. In further analysis, we noticed that all ten of these "outlier" data points for the two systems occurred when the "shot" was at node two. One likely explanation for these outliers is if the "shot" delivered

[12]Arrival time estimates were chosen using instructions from an expert geophysicist.

adjacent to node two had a less clear-cut seismic waveform than the other "shots", making arrival time picking imprecise. In future seismic refraction surveys, more data points per shot-receiver pair may help alleviate such inconsistencies. For spacing of 80 feet, the possible causes of arrival time differences is discussed in Section 5.4.3.

To quantify the variance between systems, we evaluated the differences in estimated arrival times for the wired and wireless systems. Specifically, we calculated the root mean square difference (RMSD) between each pair of shot-receiver data points. RMSD is calculated as:

$$RMSD = \sqrt{(w_{sr} - g_{sr})^2},$$

where w_{sr} is the arrival time estimated for "shot" s and receiver r acquired by the wired system and g_{sr} is the arrival time estimated for "shot" s and receiver r acquired by the GeoMoteShield system. Figure 5.19 shows the RMSD results for each set of arrival times given the spacing between geophones.

The most notable trend in Figure 5.19 is that, for 60 feet geophone spacing or less, almost all the arrival time differences were less than 0.005 seconds, or 5 milliseconds. At 70 and 80 feet geophone spacing, the median arrival time difference was between 10 and 15 milliseconds. In the next section, we discuss reasons *why* such differences in arrival times are high for longer shot-receive distances. In short, we believe the gain setting for the GeoMoteShields is sub-optimal for long distance (70 feet or greater) between shot-receiver pairs.

5.4.3 Seismic Velocity

Seismic velocity is correlated with different geology structures and is an important metric for geophysicists. Additionally, time of arrival information can be used to infer, based on the node placement geometry, the origin of the seismic signal.

The second step in our analysis was to calculate the seismic velocities of the subsurface using the arrival times estimated from both the wired and wireless systems. To calculate

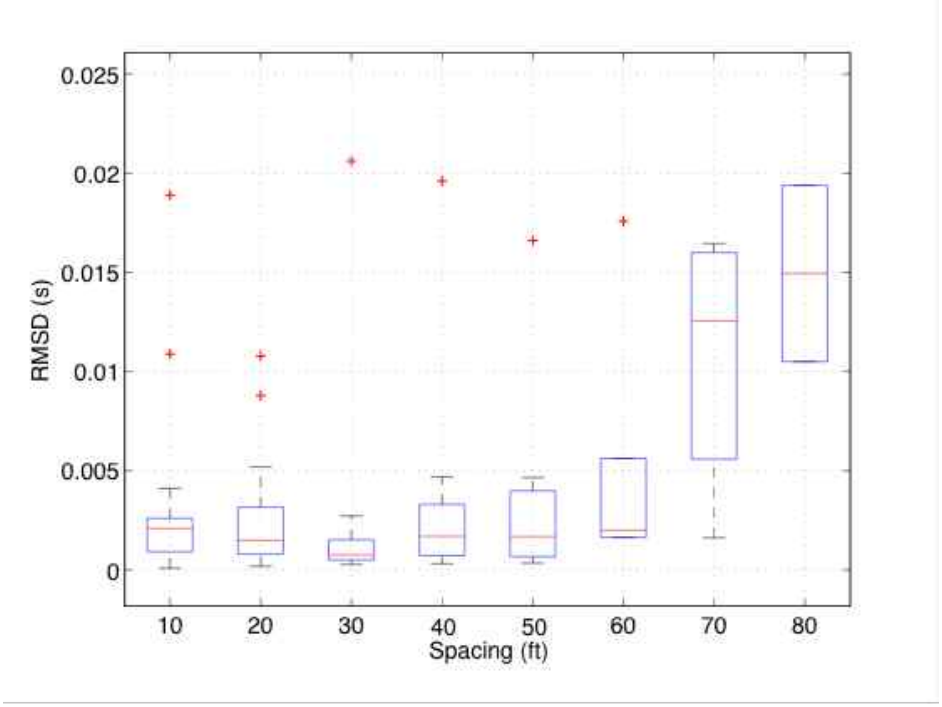


Figure 5.19: Box plots showing the RMSD values calculated between the wired and wireless shot-receiver pairs. For example, wired data recorded on node nine during "shot" one was compared to the wireless data recorded by node nine during "shot" one.

velocity (i.e., feet per second), we divided the distance between each shot-receiver pair (in feet) by the corresponding arrival time (in seconds). Figure 5.20 provides a scatterplot of the estimated seismic velocities given the distance between each shot-receiver pair. We note that one outlier from the wired system was omitted from our analysis due to an implausible velocity estimate, i.e., nearly 10,000 ft/s estimated velocity between shot-receiver pair two and three; this outlier was most likely caused by imprecise arrival time selection due to relatively lower quality (noisier) data.

To quantify the differences in estimated seismic velocities, we calculated the RMSD between the velocities estimated by the wired and wireless systems. Figure 5.21 provides statistical summaries of the differences in estimated velocities given the distance between shot-receiver pairs.

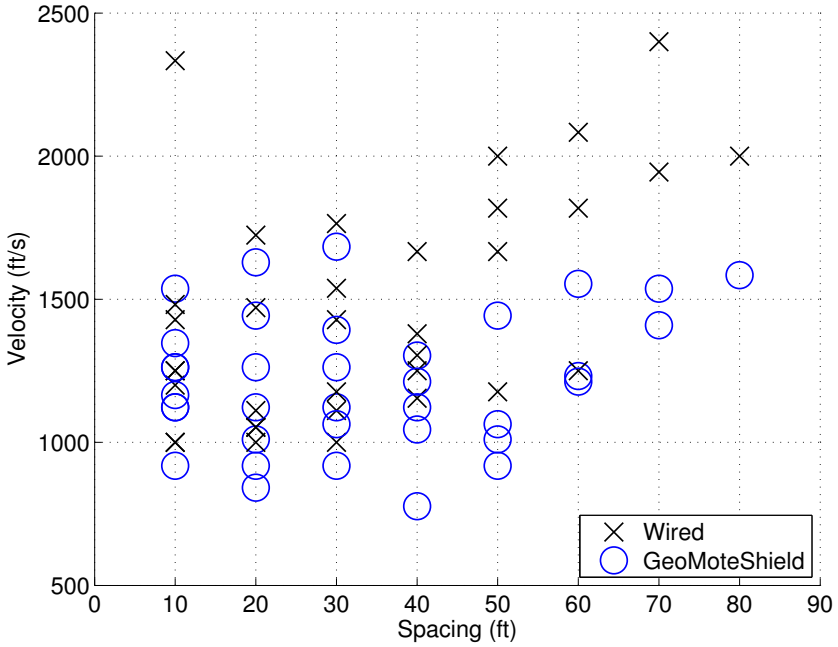


Figure 5.20: The seismic velocities estimated between all shot-receiver pairs for the wired and wireless systems.

The results in Figures 5.20 and 5.21 show that, at 60 feet spacing or less, our WSN of GeoMoteShields performs comparably to the wired system. At distances of 70 and 80 feet, however, our WSN acquired data that produced (in general) slower velocity estimates compared to the wired system. This discrepancy between estimated velocities at farther distances is likely due to sub-optimal GeoMoteShield gain settings for long distance shot-receiver pairs.

As mentioned previously, we set the gain for all GeoMoteShields to 32 (out of 128) based on previous field testing and geophysics expert opinion. It could be the case that this gain setting works best for shot-receiver pairs of 30 feet or less. At 40 feet or beyond, this gain setting may not provide an ideal signal to noise ratio for picking arrival times. In this case, the arrival times of longer distance shot-receiver pairs may be imprecise due to increased difficulty in manually picking arrivals given higher signal noise. Perhaps our WSN needs a different gain setting for each node, depending on the shot-receiver distance.

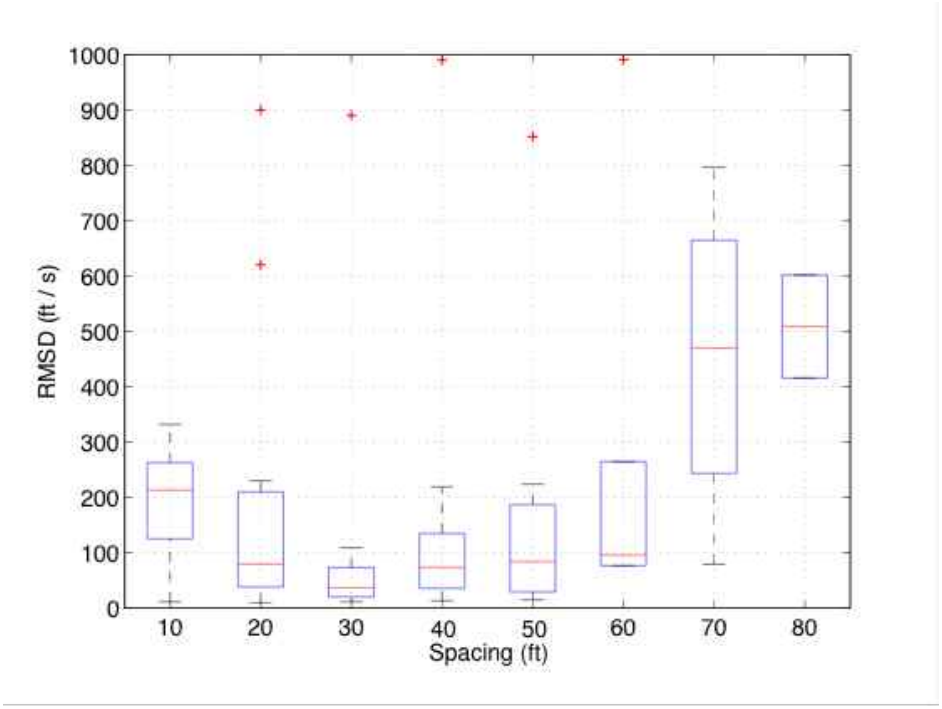


Figure 5.21: Box plots showing the root mean square difference between the seismic velocities estimated from data acquired by the wired and wireless system.

Additionally, our WSN could be designed to feature dynamic gain, where the GeoMoteShield auto-adjusts according to the input signal range. Such customization requires further field testing and calibration based on geophysical expert opinion. We note that the Geometrics Geode seismograph we used is a professional grade *product* that has likely been tested and iterated on for years. A similar amount of time and effort is likely required to bring a WSN of GeoMoteShields to market.

We note, however, that the data collected was similar with regard to the overall trends. For example, the average seismic velocity estimated from data acquired by the wired and wireless systems was 1344.6 ft/s and 1232.9 ft/s, respectively. Specifically, Figure 5.22 shows a statistical summary of the estimated velocities recorded by each system. Note the amount of overlap between the 25^{th} and 75^{th} percentiles. We also note that the wired data has a larger range because of higher estimated velocities at 50 feet offset or greater. Analysis of *why* this increased velocity occurs at higher distances is beyond the scope of this dissertation.

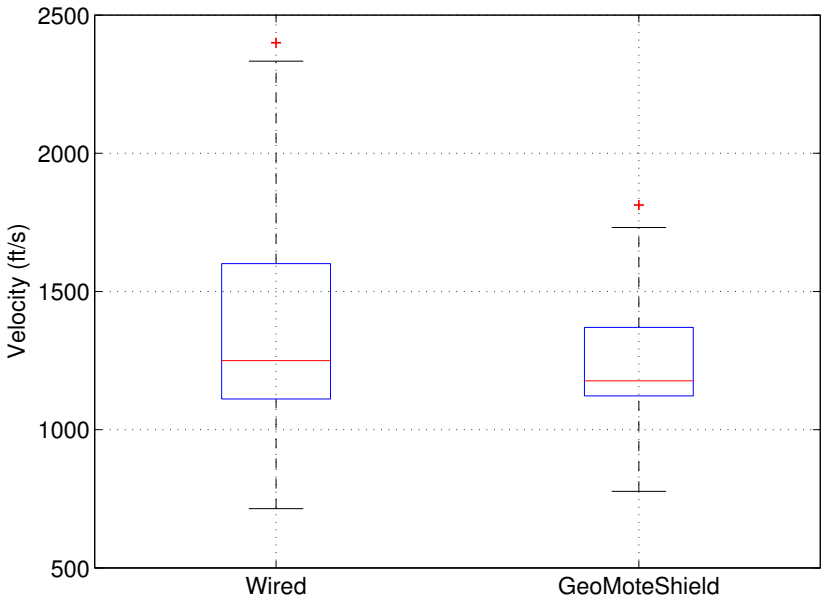


Figure 5.22: Box plots showing the statistical summary of all seismic velocities estimated for each system.

5.5 Conclusions

The results presented in this chapter show that our low-cost, Arduino-based system performs comparably to an expensive, traditional *wired* system and a fully custom (i.e., "from the ground up") wireless platform. Roughly, the Geometrics Geode seismograph costs tens of thousands of dollars (excluding geophones), each gsMote costs about $750, and our Arduino Fio plus custom GeoMoteShield "Slim" costs about $150. We conclude that Arduino should not be overlooked as a research grade WSN tool; Arduino-based systems are flexible, inexpensive, adaptable, and extremely easy to use.

There are many next steps worth pursuing, given additional time, functional equipment, and geophysics expert availability. The first next step would be to collect much more field data and have an expert geophysicist work with the seismic data (including, for example, a full inversion to image the subsurface). A second next step would be to implement dynamic gain or carefully calibrate the GeoMoteShields given the geometry of the seismic survey.

A third next step would be to perform a controlled, lab-based sensitivity analysis of the wired and wireless systems using geophysical sensor calibration methods. For example, we could compare how well the systems record a geophone's voltage output as it moves on a "shake table" at known velocities. Such information would also help us calibrate the GeoMoteShields for future field use. Lastly, a fourth next step would be to fully engineer a complete WSN. This step would involve implementing fault tolerance and "hand shaking" to check time synchronization, acknowledge commands, and verify received data quality.

CHAPTER 6

SCIENCE, TECHNOLOGY, ENGINEERING, AND POLICY: HOW, WHEN, AND WHY DOES SCIENCE MATTER IN GEOHAZARD RISK SCENARIOS?

In this chapter, I review and discuss topics and/or concepts from the broad and rich field of Science, Technology, Engineering, and Policy (STEP) that have altered my perspectives on what it means to be a scientist in society. In particular, I start by discussing two concepts, i.e., Latour's Actor-Network Theory [50] and Sarewitz's "Myth of Infinite Benefit" [51], which have changed my views of computer science and academic research. These concepts led me to ask questions regarding how, when, and why scientific work matters, *especially* in large-scale geohazardous risk scenarios.

To answer such questions, I survey literature from three fields within STEP, i.e., 1) risk, scientific uncertainty, and policy (Section 6.2), 2) society's perceptions of risk (Section 6.3), and 3) the effectiveness of risk communication (Section 6.4). Specifically, I summarize a subset of the STEP literatures to evaluate: 1) when scientific work influences policy decisions, 2) how scientific work plays into risk perception, and 3) how scientific work is communicated to the non-scientific community. I note that this chapter is *not* intended to be a comprehensive STEP literature survey; such work is beyond the scope of this dissertation. Instead, I focus on literature that is relevant to scientists in the field of geohazard monitoring who seek to, perhaps, have their technologies adopted into societal use. The questions raised and literature surveyed should help guide geoscientists to know when, how, and why their scientific work matters (and also when it does not matter).

I conclude in Section 6.5 with a discussion of three broad themes that can be drawn from the selected STEP literature. The themes are framed from the perspective of a scientist doing geohazard monitoring research and reflect how STEP concepts have influenced my thoughts on what it means to be a scientist with regard to policy, the public, and communication.

6.1 Introduction

Before delving into such questions and literature, I start by describing two concepts from STEP that have drastically altered my perspectives of what a scientist is and how science works. These concepts led me to ponder how, when, and why science is used in geohazard risk scenarios. The first concept that has altered my perspectives is Bruno Latour's "Actor-Network Theory" (ANT) [50]. This influential work revealed that science is much more of a social process than I previously believed; science and research greatly depends on the complex interactions of actors in a massive, often imperceptible, network.

The best way to explain ANT is via an example. Latour used the concept of a "black box" to help illustrate ANT; once you open Pandora's black box and begin asking questions, a complex network of actors (people, things, concepts) becomes illuminated. Take, for example, the wireless mote used throughout this dissertation. The hardware itself, i.e., the printed circuit board (PCB) and components are all *things* (actors) that come from a multitude of vendors containing different people (more actors). The network of goods, services, and intellect required to design, manufacture, and sell the hardware forms a complex network of locations, raw materials, people, and processes (each of which have their own huge network). Each individual component of the wireless mote, from resistor to radio, is also the culmination of decades of research and development from an inordinate number of scientists and engineers.

In addition to the wireless hardware, there is also mote software. This software includes the object oriented programming language, compiler, integrated developer environment (IDE), and open-source software libraries. In turn, each of these elements can be seen as their own black box, which, when opened, reveals a very large and complex network of actors (people and things) interacting in many different ways. Suffice it to say, the network attached to one wireless mote is quite massive.

ANT suggests that scientific "truths" are defined by the size, reputation, and extensibility of its network. This idea reveals a side of science that I had never considered before. Reading

and thinking about (ANT) changed the way I view the work of an academic computer scientist; specifically, ANT explains that the impacts of research are not entirely based on its technical merit, but more on the network of people and artifacts that support, use, and acknowledge such work.

To put this concept another way, becoming a bona fide academic in computer science requires more than just awe-inspiring research and publishable results. To use the common colloquialism, becoming a tenured faculty member "takes a village." This "village" consists of both people and things, including advisor(s), undergraduate students, conference reviewers, established technologies, open-source software, and other scientists. Being aware of this network of actors is important because it reveals the multitude of social interactions that are required for successful research. In a field such as computer science, where the stereotype is often a lonely programmer, this level of societal involvement may not be a common perspective; ANT shows us that we must pay attention to not only ways that technical fields can shape society, but also how social aspects can shape technical fields.

This idea, i.e., that social interactions play an integral role in technical fields, should influence the way computer scientists are educated both before and during employment. For example, developing a computer scientist's presentation skills would be a somewhat obvious way to improve their ability to communicate technical information to non computer scientists. A less obvious exercise, for example, would be to have computer scientists partake in improvisation workshops to help break the "lonely programmer in a dark basement" stereotype and learn new styles of communication and interaction. ANT shows us that social and communication skills are indirectly critical to the usage and growth of technology; hence why such skills *should* be included within any computer science education or employment track.

Before learning about ANT, I did not fully appreciate how critical diverse networks of actors are to success as an academic computer scientist; research is often only a small cog in a much bigger machine. ANT reveals that the multitude of actors within scientific work are

connected in many expected and, perhaps more importantly, unexpected ways. For example, one could make predictions regarding the expected network of actors required for academic publications; there are, for example, students, advisors, reviewers, editors, and conference committee members. Such network interactions are to be expected.

On the other hand, there are networks of actors that are less expected or unexpected. For example, much of the seismic signal processing and feature extraction used throughout this dissertation comes from the world of music signal processing. Did the scientists who created such music signal processing techniques and open-source software expect that their science and technology would one day assist computer scientists in seismic event classification? In this instance, the network of actors using scientific work from music signal processing has been expanded to include those in automated geohazard monitoring; scientific work does appear to matter, the degree of which depends on complex actor-network interactions and not the level of "truth" of the scientific work. Contemplating this perspective may help geohazard scientists navigate having their scientific work used to predict or mitigate a geohazard risk. Put another way, science is much more of a social process than purist scientists are trained to believe; the social aspects of geohazard risk prediction or mitigation cannot be ignored due to their unscientific nature.

The second concept from the STEP literature that has shifted my perspectives and led me to ask questions about the role of scientific work is Sarewitz's "myth of infinite benefit" [51]. In summary, Sarewitz claims that more and more science does not necessarily equate to better society or quality of life. I expand further on this provocative claim below, but the significance is that this pragmatic outlook has altered my views regarding *when* and *how* scientific work does and does *not* matter in determining societal outcomes.

Sarewitz provides a compelling and provocative argument against the often used "myth of infinite benefit", the notion that more scientific research equates to better life for society. The author cites several examples of how funding more research and technology development has not necessary led to better quality of life for society. Instead, Sarewitz promotes the

use of science for science policy, where decisions regarding federal spending on research and development incorporate the societal, cultural, and economic effects in addition to the scientific benefits. The author also suggests that an ever-increasing science and technology front is unsustainable, and instead funding agencies should focus on how to understand and improve the workflow between basic and applied scientific research, technology innovation, acceptance, and societal improvement.

Sarewitz offers several succinct examples of how the "myth of infinite benefit" is incorrect. The most compelling evidence the author provides is the USA's poor health despite the massive growth of medical research and development. In this example, Sarewitz points out that, although the USA spends 10 times more money than Germany and Japan on medical research, the USA falls short in terms of basic health indicators such as infant mortality rates and life expectancy. Additionally, the author states that healthcare in the USA can only be afforded by a privileged few, which further questions the societal benefits of extensive and ever-growing medical research and development. One reason why this view is so strong is because it is as true today (2014) as it was in 1996, when Sarewitz's book was published. The USA still lags behind much of the industrialized world in terms of health indicators (e.g., obesity rates) despite our technologically advanced medicine. In the case of medicine in the USA, there are many other aspects, aside from science and technology, that determine people's health status.

Sarewitz seems to instruct scientists to be pragmatic when thinking about how their science will impact society. The realized societal outcomes are based on much more than scientific consensus or trends in technology innovation. This concept has led me to a much larger question regarding when scientific work actually *does* and *does not* matter for societal outcomes. Specifically, the "Myth of Infinite Benefit" struck close to home; the geohazard monitoring research presented throughout this dissertation is motivated by improving societal outcomes, yet it is unclear whether the results presented herein will lead directly to positive societal impacts (e.g., a safer public).

So, in the area of geohazard risks, does science matter? How and when does scientific work influence policy decisions? How does scientific work play into the public's perception of geohazard scale risk? When is scientific work communicated to the public and is it effective? In the literature survey and analysis that follows, I attempt to answer such questions and learn what it means to be a scientist involved in geohazard monitoring.

6.2 Risk, Scientific Uncertainty, and Policy

Before completing the STEP minor, I had an unsophisticated perspective on how science informed decision making, especially with regard to environmental risks. Typical of an engineer, I naively believed that science *should be* the deciding factor in informing risk behavior and policy. When science produced inconclusive results regarding risk outcomes, I would have called for "more science" (following the "myth of infinite benefit" mentality discussed previously).

After reviewing a portion of the STEP literature in the fields of risk, scientific uncertainty, and policy (as it relates to geohazard risks), I now have a more nuanced understanding of, and appreciation for, decisions in light of scientific uncertainty; the vast majority of policy decisions are made from imperfect knowledge. In addition, I also note that policy decisions are just as often made on value judgements than on scientific "knowledge", especially in risk scenarios that involve a high level of controversy. As such, my perspectives on the authority of scientific expertise have become less rigid; the questions of "how" and "why" science matters in policy decisions is quite a gray-area subject without clear cut answers.

In this section I summarize and analyze literature regarding the complex interrelationships of risk, scientific uncertainty, and policy. I note that the literature surveyed is limited to articles related to large-scale, geohazardous risk, where the risk may not be obvious or overtly apparent to a non-scientist; the scope of literature was selected in an attempt to stay consistent with the topic of this dissertation (i.e., geohazard monitoring). In other words, I only review and discuss STEP articles that are closely related to large-scale risk mitigation; going deep into each STEP concept is beyond the scope of this chapter.

Suffice it to say, the intersection of risk, scientific uncertainty, and policy is complex and dynamic. Scientific uncertainty will always be part of "science-based" policy decision making, especially when risk is involved. As I attempt to show using the following selection of literature however, scientific work is but one aspect considered during policy decisions. Even if there was zero scientific uncertainty and perfect consensus, risk policy decisions would inevitably include aspects from many expected and unexpected actors. Risk policy decisions require weighing not only the science, but also political, social, and economic aspects as well. Though science plays an important role in policy decisions, it is not the only factor used to persuade or dissuade policy makers. In essence, science *does* matter, but it is not the *only* factor.

To begin with, scientific work often contains levels of uncertainty. Unfortunately, scientific uncertainty has and can be blatantly used to hinder policy decisions; politicians have learned to wield the "more science is needed" mantra to delay action in spite of scientific consensus. Oreskes and Conway cite countless examples of this phenomenon, e.g., second-hand smoke, acid rain, climate change, chlorofluorocarbons (CFCs), and dichlorodiphenyltrichloroethane (DDT) [52]. In each case, the actors opposed to regulation use scientific uncertainty to delay or kill regulation; science is *not* the only or most important factor.

For example, during the 1980s, "doubt-mongering" was used to slow regulation and policy action regarding acid rain [52]. Regulation policies were significantly delayed because opponents successfully argued that "the science was too uncertain to justify action" and action was economically infeasible. In other words, opponents of regulation made a concerted effort to discount the certainty of scientific consensus, and did so by any means necessary. Should policy makers wait for 100% scientific certainty and consensus before making new regulations? If not, what is an "acceptable" level of scientific uncertainty? When does scientific uncertainty truly matter and when can it be ignored?

Downton et al. analyze and discuss two flood management case studies in Colorado that highlight how uncertainty can drastically influence policy decisions and outcomes [53]. The

first case involves redrawing floodplain maps in Fort Collins, and the second case concerns raising the height and safety of the Cherry Creek Dam in Denver. In each case, the authors point out how scientific uncertainty was used to either support new flood regulations (i.e., Fort Collins) or to refute costly dam construction (i.e., Cherry Creek Dam).

In 1997, a massive flood in Fort Collins caused catastrophic damage to areas that were not within the 500-year floodplain areas developed in the 1980s. In light of this, the local government called for a major overhaul of their flood management policies, including a comparative scientific investigation of area rainfall rates and frequencies. The results showed that previous floodplain maps, drawn based on data from the National Oceanic and Atmospheric Association (NOAA), were faulty because of the uncertainty inherent in using sparse historical records to make flood predictions. Instead, scientists collected, analyzed, and developed more precise (yet still uncertain) flood models based on rain gauge data from Fort Collins and surrounding areas. Using this new evidence, Fort Collins City Council redrew flood maps and raised design standards of the city's stormwater handling infrastructure (e.g., drains, culverts, and bridges). In this case, the scientific uncertainties of "old" floodplain models was brought into question and used to motivate new policy decisions. Downton et al. note, however, that the uncertainty of the modern regional model was not called into question; instead, policymakers accepted the updated model as absolute truth.

In the second case, scientific uncertainties played a much bigger role in deciding the fate of the flood management issue raised. In the 1990s, the U.S. Army Corps of Engineers (USACE) deemed that the Cherry Creek Dam, designed based on flood estimates from 1944, could only control 63% of probable maximal flow[13] and was thus deemed unsafe. Representatives from USACE noted that, though the probability of catastrophic dam failure was remote, the consequences would be quite dire, with estimated damages in the tens of billions of dollars. The projections made by USACE, based on precipitation models from the National Weather Service (NWS), were called into question by local, regional, and state

[13]Probable maximal flow is a metric commonly used by dam engineers to estimate how much water a dam can safely control.

governing bodies. Thus, the Colorado Water Conservation Board (CWCB) hired a local consulting firm to conduct a peer review of the NWS based model of precipitation and resulting dam performance. Review results suggested that the NWS based precipitation model was 25% too high, implying that the projections made by USACE regarding dam safety were unrealistically negative. Additionally, the new estimates suggested that the dam was still within federal regulations, and thus no repairs were necessary. In this case, scientific uncertainties and disagreement associated with precipitation models allowed policymakers to refute costly dam repairs.

These two case studies discussed by Downton et al. show how scientific uncertainty can drastically alter the decisions and directions of policymakers. Though the two cases were similar in terms of meteorological model uncertainty and flood management, the way uncertainty was framed had very different consequences. On the one hand, the city of Fort Collins now has improved flood management infrastructure to better deal with catastrophic precipitation rates. On the other hand, the refusal to improve the Cherry Creek Dam may someday cost downtown Denver billions of dollars in damage from an overtopping dam failure. Clearly, the questions of how and when scientific work matters in policy decision is quite a complex phenomenon; the same scientific work can be used to both support and refute policy decisions.

Should scientific work then be relegated only to the experts? Does the nonscientific public "need to know" the nuances (e.g., uncertainties) of the science involved? In a historical account of seismology in California, Meltsner shows how scientific information can be suppressed for the "public good" [54]. Meltsner also discusses how scientific uncertainty can be problematic when it involves public affairs; he concludes that sometimes science should remain in the laboratory and the public should simply not know about it.

The first case regarding the suppression of scientific information came after the 1906 earthquake in San Francisco. The 1906 earthquake was very destructive, with fires that decimated much of San Francisco. Fearing that reports of an "earthquake" would damage

California's reputation and decrease business, newspaper articles and radio broadcasts failed to mention that the damages were caused by an earthquake. Instead, the media proclaimed that all damages were caused by fire. This "chamber of commerce" mentality continued for several decades; politicians wanted nothing to do with policies regarding earthquake preparedness, the public did not want to hear about earthquakes, and scientists could not readily acquire funds for seismic research.

Attitudes shifted in 1925, when a catastrophic earthquake hit Santa Barbara. After this earthquake, a seismologist named Bailey Willis began to communicate information regarding earthquake preparedness to many different types of groups, including insurance underwriters, business owners, builders, and architects. Willis' message was focused on the economics of designing and building earthquake resistant buildings. Though Willis did not get the building codes he wanted put into law, he was successful at increasing earthquake insurance rates.

Although the Seismological Society of America published a fault map of California in 1923, governmental maps did not include these faults until 1938. The government did not want to assume liability in deciding where (or where not) to build new structures. This delay helps show the complexities regarding perceived scientific uncertainty that Meltsner alludes to. Specifically, the non-scientific public (including governmental officials) preferred an "ostrich policy" regarding earthquake preparedness; rather than confront the problem of earthquake susceptibility with new and uncertain science, the non-scientific public preferred to stick their collective heads in the sand, hoping that the problem would simply go away. The historical accounts presented in Meltsner's article remind scientists that uncertainty can be used as rationale for inaction.

The case studies presented from Colorado [53] and California [54] serve to remind scientists that, again, their scientific work may or may not be included in the policymaking decision process. Even if their scientific work is included, researchers may have little control over *how* their scientific work is utilized or *why* policymakers choose to base their decisions on certain scientific results over others. The take home message reflects the complexities of

policymaking for risk scenarios, especially large-scale geohazard risks. Researchers in geohazard monitoring should acknowledge that, at some point, the use of their scientific evidence is beyond their immediate control. Note that this observation relates to ANT [50] discussed in Section 6.1; the dissemination and use of science is based more on a complex network of expected and unexpected actors than the degree of "truth" or "fact" of the scientific work.

Marcus states that scientists who advise policymakers are faced with a troublesome dilemma: scientists must make advisements that are based on science yet, paradoxically, cannot be answered with perfect certainty by science [55]. Science is inherently uncertain and all of science lies somewhere between perfect knowledge (100% certainty) and perfect ignorance (0% certainty). Even if 100% certainty about some physical system existed, it would not solve the political, social, and economic problems of weighing the anticipated costs and projected benefits. Thus, believing that political processes can be neatly resolved with more science is an oversimplification.

Marcus notes an interesting duality between scientific and economic uncertainty when it comes to acceptable risk. For example, uncertainty in economics is nothing new; entrepreneurs constantly deal with uncertainty. Yet, if and when entrepreneurs fail, they are not held to the same standards as when scientists falter. In other words, when an entrepreneur fails, the institution of entrepreneurship and economics is not harmed. However, when scientists fail to make accurate predictions regarding risk (even in the face of uncertainty), the institution of science is often degraded. This begs the question: *why* is scientific work held to a different standard during policy decision making?

The difference between science and economic risk, as Marcus claims, is that scientists cannot allocate risk to only certain groups of people. When entrepreneurs take risks, the risk is allocated to only a select few, e.g., the entrepreneur, investors, employees, and manufacturers; however, when scientists make predictions pertaining to risks on a larger scale (e.g., nuclear power plant fallout, second hand smoke, ozone depletion, and climate change), they cannot allocate the risk to a particular group of people. Thus, since scientists cannot

focus risk and the associated uncertainties of that risk onto a subset of the population, they are held to a different, higher standard because they are responsible for guarding society as a whole.

Even if risk could be focused to a small, highly localized population, how would the non-scientific public respond to these pinpointed risk assessments? In risk psychology research, people are more likely to fear large events of low probability (e.g., nuclear fallout) than highly probably events of relatively little consequence (e.g., riding a motorcycle). The result of this phenomenon is that explaining probabilities to the non-scientific public may be of little value, as people's preconceived notions of risk and uncertainty are hard to overcome. In other words, probabilistic science may not satisfy the demand of policymakers for precise answers with regard to risk and uncertainty. Furthermore, Marcus shows how governmental agencies and private companies deal with risk in very different ways. There is simply no magic bullet explaining how scientists should advise policymakers or authority figures in matters of risk. In this regard, Marcus states that scientists must develop intuition on how to make or advise decisions in the face of complex and uncertain science.

The problem seems to be that scientists and engineers are trained in well controlled labs with binary, yes/no, black/white questions and answers. In the real world, however, policy decisions involving complex systems require answering questions that go beyond mathematic equations or rule based algorithms. The ability to answer more ambiguous questions in the "grey" areas requires skills that are not taught during a scientist or engineer's training. Developing these "informal rules", as Marcus denotes, is similar to how physicians learn to make diagnoses despite imperfect knowledge. In light of this, Marcus claims that such "clinical" skills should be taught as part of a scientist or engineer's training, much like how physicians spend years building intuition during medical school, internships, and residencies. In addition, scientists and engineers must embrace error and learn from previous mistakes.

Perhaps it is the case that scientists and engineers simply need more clear-cut definitions of different types of uncertainty. Ascher attempts to define, rank, categorize, and explain

different types of scientific uncertainty; efforts that may help scientists deal with inherent scientific uncertainty. The author asserts that scientists must guard against the notion of uncertainty and inaction, or that "more research is needed" [56]. Instead, Ascher provides a taxonomy of uncertainty and lists ways for scientists to become better involved with policymaking decisions and action. The "more research is needed" sentiment has been seen in the case of climate science, where climate model uncertainties have been used as rationale to ignore scientific consensus. In other words, policymakers often claim that the model uncertainties must be resolved before action is taken. At the same time, however, scientists cannot exaggerate certainty; uncertainty is inherent in science. Thus, scientists are stuck in a paradox: in essence, scientists must be able to say, with a high degree of confidence, how much they do not know.

To help address this paradox, Ascher defines a taxonomy of uncertainty, where different levels of uncertainty can be better elaborated. At the highest level, there are two types of uncertainty: epistemic and aleatoric. Epistemic uncertainty pertains to incomplete knowledge about the physical systems scientists try to model. This epistemic uncertainty can be classified into five subcategories: 1) laws of nature, 2) subtle-effects, 3) state of nature, 4) parametric uncertainty, or 5) incalculability. Table 6.1 further defines each subtype of epistemic uncertainty.

Table 6.1: Subcategories of epistemic uncertainty defined by Ascher (2004).

Subcategory	Definition
laws of nature	ignorance about system behavior
subtle-effects	inability to model small effects of a large system
state of nature	ignorance about past or present conditions
parametric uncertainty	ignorance about the parameters or weights to be assigned to system processes
incalculability	impossible to calculate outcomes based on system complexities and multiplicity of interactions

Aleatoric uncertainty, which is the uncertainty inherent in stochastic or random systems, can be further classified into three subcategories: 1) intrinsic, nano-level, 2) non-model, or

3) application-case. Table 6.2 defines each subtype of aleatoric uncertainty.

Table 6.2: Subcategories of aleatoric uncertainty defined by Ascher (2004).

Subcategory	Definition
intrinsic, nano-level	due to truly unknowable or unpredictable factors
non-model	factors excluded from the model (usually at the micro level)
application-case	when actual cases do not fit the current model

Ascher gives a list of actions that scientists can take to avoid the misuse of scientific uncertainty. First, scientists must be explicit about the types of uncertainties found in their science. The taxonomy of uncertainties that Ascher defines provides a language that may aid communication. Second, scientists must show good faith efforts to reduce uncertainty and increase credibility. This assumes, perhaps incorrectly, that science has the capacity to eventually resolve most (if not all) uncertainty. Third, scientists should make provisional consensus agreements that focus on what *is known*. Fourth, scientists must strive to get funding from a diversity of agencies, and not just, for example, the Environmental Protection Agency (EPA). Lastly, scientists must include the status quo in their projections, to provide policymakers with an alternative.

In addition to more clear-cut definitions of uncertainty provided by Ascher, scientists and engineers may benefit greatly from better definitions with regard to the "amount" of certainty required to make substantive claims. Weiss argues that, when science is used as reasoning in policy and regulatory decisions, scientific evidence (including uncertainty) should be defined in terms of the "standard of proof" in the United States legal system [57]. This notion of establishing different levels of evidence is well documented in the U.S. legal system, but the same cannot be said for the level of scientific proof necessary to make precautionary regulations against hazards that *may* cause severe and irreversible harm to the environment. In this regard, Weiss analyzes several case studies to better understand how the standard and burden of proof can be defined when it comes to environmental hazard regulation.

In each case study evaluated, Weiss analyzes who the burden of proof falls on and shows how the standard of proof depends ultimately on who has the burden of proof. For example, those enlisting the precautionary principle, which states "action to protect the environment against danger of severe and irreversible damage need not wait for rigorous scientific proof" [57, p. 139], only need to reach the "reasonable belief" standard of evidence to make a solid case (10-33% certainty). On the other hand, proponents for actions that *may* cause severe and irreversible damage must reach a higher standard of proof akin to a "clear showing" in court (about 80-90% certainty).

To help clarify the different levels of uncertainty, Weiss provides four scales of uncertainty levels and relates them to each other. Specifically, Weiss compares the Intergovernmental Panel on Climate Change's (IPCC) 7-point qualitative scale of uncertainty to the 11 point scale regarding the standard of proof in U.S. law (see Table 6.3). Of course, Weiss states that his definitions of the burden and standard of proof are mostly conceptual, and does not take other factors (e.g., economic, political, religious, social, and cultural factors) into account. In summary, Weiss's main contribution is a framework and language with which policymakers can negotiate regulations by treating scientific evidence with the same standards of proof that are required in U.S. courtrooms.

Table 6.3: The relationship between the IPCC's qualitative scale of scientific uncertainty and the standard of proof used in U.S. law (Weiss, 2003).

IPCC scale	Standards of Proof (U.S. law)	Certainty (%)
(not in scale)	beyond any doubt	100%
virtually certain	beyond a reasonable doubt	99%
very likely	clear and convincing	90-99%
likely	clear showing	80-90%
likely	substantial and credible	67-80%
medium likelihood	preponderance of evidence	50-67%
medium likelihood	clear indication	33-50%
unlikely	probable cause; reasonable belief	10-33%
unlikely	reasonable grounds for suspicion	1-10%
very unlikely	inchoate hunch	<1%
(not in scale)	impossible	0%

The work by Marcus, Ascher, and Weiss shows *how* scientific work is scrutinized during the policymaking process. The author's contributions seem to instruct scientists to be cognizant of the uncertainties in their scientific work, especially when such work is used to help make forward-looking policy decisions based on predictive models. Thus, scientists in geohazard monitoring research must, for example, be aware of the types of uncertainty inherent in their systems (e.g., epistemic or aleatoric) and be able to explain such uncertainties in other terms (e.g., legal definitions).

What happens when scientists fail to reveal the uncertainty of their work? In an ethical assessment of earthquake prediction, the burden of proof and explanation of scientific uncertainty was clearly neglected in a case study of earthquake prediction. Sol and Turan [58] provide examples of how scientific uncertainty can play a significant role in altering the public's trust in science. One example case study is the Brady-Spence prediction of 1981. During that year, Brady, a scientist with the United States Bureau of Mines, and Spence, a geologist with the United States Geological Survey (USGS), predicted that a series of catastrophic earthquakes would hit Peru between June and September, 1981. This information was leaked to Giesecke, director of the Geophysical Institute of Peru, who chose not to share this information with the Peruvian public. In other words, Giesecke and other Peruvian officials did not want to inform the public because they did not want to cause an economic downturn due to decreased tourism and business investments. At the same time, however, Brady and Spence gave interviews in the United States where they shared their predictions of the upcoming Peruvian earthquakes. The information shared in these interviews caused significant public anxiety in Peru, where thousands of people made preparations for catastrophe and evacuated prematurely for an earthquake that never came.

In this case, the scientific uncertainty of earthquake prediction was *not* shared explicitly with the Peruvian public. The Peruvian authorities chose not to share the earthquake predictions at all, in order to avoid unnecessary social disturbances; however, by not clearly conveying the predictive science and associated scientific uncertainty, the Peruvian officials

created an anxiety ridden environment that they had been trying to avoid. Sol and Turan claim that the failure to inform the public on the unreliability and scientific uncertainty of earthquake prediction is unethical; in short, seismologists have a responsibility to explain the uncertainties of earthquake prediction.

The case presented by Sol and Turan is contrasted by recent events in L'Aquila, Italy, where six seismologists and one public official were convicted of manslaughter and sentenced to prison for failing to adequately evaluate and communicate the risk of a potential earthquake [59]. On April 6^{th}, 2009, a 6.3 magnitude earthquake hit the city of L'Aquila, Italy, killing 300+ people and causing massive structural damage. In the days and weeks prior to this massive earthquake, the small city experienced hundreds of smaller seismic tremors, raising fears of the public. On March 31^{st}, 2009, just days before the deadly earthquake, seismic officials met to discuss the risk of a major earthquake. After the meeting, these government officials held a press conference where they reassured the public by saying that the small tremors did not increase the risk of a large, catastrophic earthquake. However, a massive and destructive earthquake did befall L'Aquila about a week later. The defendants in this famous case claim that the seismologists failed to "adequately evaluate, and then communicate, the potential risk to the local population" [60]. In other words, the seismologists convicted were not being blamed for failing to *predict* the exact location, time, and magnitude of the earthquake, but instead were on trial for not effectively communicating the risk inherent of living in an active and dangerous seismic region.

The Brady-Spence prediction and story of the convicted Italian seismologists touch upon two other important aspects in regard to how scientific work matters in geohazard risk scenarios. The first question is: to what degree does scientific evidence play a role in society's perception of risks (the topic of Section 6.3)? If the Italian seismologists could have predicted that a massive earthquake would hit the area on April 5^{th}, a day before the actual earthquake, would the public have taken appropriate actions? The second aspect is the importance of effective risk communication, especially with regard to large-scale risks (the topic of Section

6.4). What should have the Italian seismologists done differently to properly warn the citizens of L'Aquila and avoid prosecution?

6.3 Society's Perceptions of Risk

Given that policy makers must grapple with scientific uncertainty in risk assessment, how does a more general "society" perceive risk in light of uncertainty? How does scientific work play into people's perceptions of risk? In this section, I summarize literature from the field of risk perception and reflect on the authority of science. To reiterate, I focus on literature that pertains to large-scale, geohazardous risks; this chapter is not intended to be an exhaustive STEP literature survey. Instead, I select literature that is most relevant to scientists in geohazard monitoring research. In essence, the literature surveyed shows that science plays a small (if any) part in determining the public's perceptions of risk.

Johnson and Scicchitano examine the complex interactions and relationships between trust, risk, uncertainty, and the public's willingness to take action on environmental issues [61]. To accomplish this study, Johnson and Scicchitano surveyed over 400 United States citizens and asked questions regarding information, uncertainty, trust, and environmental activism. The survey was premised around two environmental risks, one that was seemingly avoidable (tap water contamination), and one that was unavoidable (nuclear power). With regard to trust, the survey included two types of informants: university scientist or government employee.

There are two sets of survey results worth mentioning. In the first set of results, survey participants were significantly more likely to trust a university scientist than a governmental official in topics related to tap water and nuclear power. Delving further, participants were significantly more likely to trust the scientific expert's opinion regarding the risks of tap water than the risks of nuclear power. In addition, there was a significant positive relationship between uncertainty and trust; i.e., as uncertainty increased, so did trust of expert opinion. Likewise, when uncertainty decreased, trust in expert opinion also decreased. Furthermore, participants were significantly more certain of the risks associated with nuclear power than

the risks associated with tap water contamination. These results suggest that, as the severity and certainty of perceived risk increases, the non-scientific public is less likely to believe the scientific expert's opinion.

The second set of results pertain to the relationship between perceived risk and activism. Survey results showed a positive relationship between the perceived risk and willingness to take action. Specifically, the greater the perceived risk, the more likely people are to take action to mitigate the risk. In addition, survey participants listed voting as the most desirable form of activism over donating money, writing a letter, or attending a local meeting.

The results of this study suggest that scientists should spend more time in non-scientific domains for building trust. In other words, the non-scientific public will not just blindly accept information from university scientists or government officials simply because they are experts. Another more pessimistic insight offered by this article is that scientists should make sure the non-scientific public knows how much they do not know. As the results show, the more certain people are about a perceived risk, the less likely they are to listen to expert opinion.

The suggestion that scientists and engineers learn to build trust with the non-scientific public speaks to a greater systematic problem within typical science, technology, engineering, and math (STEM) education. Scientists and engineers are trained in technical schools, colleges, and universities to refine particular skills specific to science and engineering; post-secondary STEM students are typically not trained in matters pertaining to "soft" skills such as, for example, communication, leadership, and conflict resolution. In other words, the system is setup to discourage learning skills to aid such trust-building interactions between scientists, engineers, and the non-scientific public.

With regard to risk, what aspects influence the non-scientific public's decision making? Kellens et al. conducted a survey to assess the public's perception of flood risk on the Belgian coast [62]. In Belgium, roughly 4% of the population (about 0.4 million people) live in flood prone areas where the potential for catastrophic coastal floods is an ever-present risk.

Additionally, during the summer tourist season, the Belgian coastal population increases by about 300,000 residential tourists. In this study, the authors distributed hundreds of surveys and collected sociodemographic information from permanent and touristic residents in three different Belgian coastal towns. The first town, Ostend, was selected because scientific experts have deemed it as a "high-risk" town, where, based on geographical features and infrastructure, coastal flooding would be especially detrimental. The other two towns in this study, i.e., Knokke-Heist and De Panne, have been deemed "low-risk" by the scientific experts. Kellens et al. hypothesized and confirmed that residents living in the "high-risk" town of Ostend will exhibit significantly higher perceived risk of coastal floods than the other two towns.

Though this result suggests that the non-scientific public *did*, to some degree, heed the implicitly uncertain risk assessments of scientific experts, further analysis revealed several mitigating effects that altered people's perspectives. Most notably, the perception of coastal flood risk was mediated by personal experience; i.e., those without direct experience with coastal floods had significantly lower risk perceptions, regardless of location.

The authors posit that the main reason why the perceived risk was higher in the "high risk" town than the "low-risk" towns was *not* because of expert assessment, but due to generations of local residents sharing stories and experiences with catastrophic coastal flood events. This explanation makes sense, since personal experience with coastal flooding was one of the most reliable predictors of perceived risk. From a more scientific point of view of risk assessment, one would expect location to be the primary factor of risk perception. Did the residents of coastal Belgium simply not trust the scientific assessment of flood risk? These results suggest that the non-scientific public's perception of risk is not neatly casual; there are often many non-scientific aspects that can drastically impact the perception of risk, regardless of scientific expert assessment.

The next example of large-scale, imperceptible risk comes from Italy in the 1970s. This case study was selected to illustrate, yet again, how and when scientific work matters in the

public's perceptions of risk. On July 10^{th}, 1976, a chemical plant located in Seveso, Italy (a suburb of Milan) accidentally released 30 kilograms of tetrachlorodibenzodioxin (TCDD) into the atmosphere [63]. Scientists at the chemical plant advised local authorities to evacuate the suburb due to harmful levels of dioxin, a chemical hazard that is toxic and deadly. On advisement of the chemical scientists, the plant was shut down and the nearly 700 residents of Seveso were evacuated to hotels in safe locations. Three months later, against the advisement of the scientists and local authorities, residents broke into the roped off areas in order to return home. In fact, it was reported that residents were seen eating lunch in their home gardens, despite being advised by officials to stay away from the area until it was deemed safe. (It is estimated that over 10,000 animals in Seveso area perished). In some cases these residents had to be removed with physical force. A post-hoc investigation showed that the chemical spill was directly linked to increased cancer rates within the Seveso population.

In the "Seveso Disaster", the residents of Seveso blatantly ignored the warnings posted by scientists (in the form of ropes and signs) in favor of returning home. Since the toxic chemical dioxin was imperceptible to residents, they were able to rationalize reasons to ignore the scientific experts. In this case, the science was *not* uncertain and the scientific work mattered a great deal. Chemical scientists at the plant knew that dioxin was harmful, and thus forced residents to evacuate; in fact, it was reported that the plant worker who noticed the leak immediately put on a gas mask and hazmat suit. Again, this case study shows that the non-scientific public's perception of risk may not be easily swayed, even in the face of scientific certainty.

These examples from the USA, Belgium, and Italy presented in this section show how and when scientific work does *not* appear to matter with regard to society's perceptions of risk; in other words, scientific work may not always be the deciding factor. In addition, this work shows that risk communication is rarely obvious or straightforward. In all cases, there were a multitude of outside factors (besides scientific assessment) that influenced how people perceived the large-scale risk from nuclear fallout, tap-water, flooding, or poisonous chemi-

cals; people's risk perceptions are determined by trust, personal experience, and perception of imminent risk. These particular studies were selected to remind scientists (working in geo-hazard monitoring research) that results from their scientific work may not lead to societal technology adoption or risk aversion.

The next article presented is another example illustrating why scientific work typically does *not* matter much in determining people's risk perceptions. In essence, the problem is that the "top-down" model of information transfer that scientists are taught is not effective. In an article published in 2010, Liverman discusses several communication barriers for scientists in conveying messages to the non-scientific public about risk and uncertainty pertaining to hazards [64]. Before discussing the communication barriers and possible solutions, Liverman first explains how and why he thinks the "top-down" model of information transfer is ineffective.

In the "top-down" model, scientific experts communicate information regarding hazard and risk to the non-scientific public. Liverman posits that this model is largely ineffective; the "I told you so" mentality simply does not work (for the most part). Liverman explains that the "top-down" model of information transfer is ineffective because the underlying notion of filling the "information deficit" is fallacious. In other words, the idea that informing the non-scientific public about risk, or filling the so-called "information deficit", would change their behavior is inaccurate; however, since geoscientists best understand the risks and predictive uncertainties associated with hazards, Liverman argues that scientists *must* still try to effectively communicate information to the non-scientific public, and let the public use that information to make decisions.

The literature presented in this section suggests that the perception of risk is a social process where personal beliefs and experiences are bigger factors than scientific work in the decision making process. In other words, one cannot assume that science alone will provide ample justification to motivate social action or policy change; the "top-down" model of information transfer is not always effective. This begs the question: why doesn't the

"top-down" model of information transfer work? I discuss risk communication in the next section.

6.4 Effective Risk Communication

In this section, I review a subset of literature from the field of risk communication. In particular, I focus on literature that pertains to large-scale, geohazardous risk, where the risk may not be obvious or overtly apparent to a non-scientist. In other words, scientific work is critical to assessing and mitigating the risk. Following is a review of articles pertaining to the question: in matters of large-scale risk, how is scientific work communicated? Additionally, how can scientists improve their ability to communicate their scientific work to help assess risk?

Effective risk communication is essential to almost every federal government agency involved in public safety. For example, the Environmental Protection Agency (EPA) must update communities about toxins at Superfund sites, the Army Corps of Engineers must convey pertinent safety information regarding levees surrounding vulnerable populations, and the Nuclear Regulatory Committee (NRC) must inform residents adjacent to nuclear power plants and nuclear waste sites about the risks thereof. In this section, I summarize and reflect on portions of the NRC's risk communication policies and guidelines [65]; I then conclude that, although most aspects of the NRC's materials are perceptive and backed by the risk communication literature, other portions are ill defined or unrealistic to implement.

In this section, I focus on the NRC because it is a well establish entity in an industry where the risk is quite polarizing, the science is complex, and, thus, effective communication of scientific work is extremely important. On one side, proponents of nuclear energy often claim that it is a safe and relatively low risk; the risks are clearly understood and there are many measures taken to minimize such risks. On the other hand, opponents of nuclear energy claim that the risks of nuclear power are extremely high and seemingly unavoidable. In the case of nuclear energy, the clear communication of scientific work regarding the risks of nuclear power is paramount to both the industry and the livelihood of the citizens living

adjacent or downwind of nuclear power plants.

Before examining the NRCs risk communication policies and guidelines, I first motivate the need for improved risk communications by the federal government. In an article written by Thomas [66], a former EPA worker, he describes his personal views regarding why risk communication is so important, especially when it involves federal government agencies managing public safety. Thomas first describes older times when the federal government relied on expert judgment to determine, in private, if something was either safe or unsafe. During this period, determining the allowable risk was not a public process, nor did the scientific experts explicitly discuss the risks. In other words, scientists decided, behind closed doors, what was safe or unsafe for the public.

Thomas goes on to describe three reasons why this system of governance is inadequate. First, technology development has fundamentally changed risk; modern risks are hard to quantify, subtle, and most likely involuntary (i.e., those involved have little to no choice in accepting the risk). Second, Thomas states that the public has changed. Today, when compared to the past, the public is less tolerant of health risks, trusts the government less, and is better informed by the plethora of available information. Lastly, the author notes that the government has changed; the government is expected to be more open, forthcoming, honest, and responsive than in previous decades. Given these three reasons, Thomas concludes that governmental agencies cannot determine acceptable risk involving public safety in a private manner and, thus, such agencies should be adept at communicating risk.

Thomas identifies several challenges that the EPA must cope with to improve risk communication. The most salient challenge is the difficulty of addressing residual risk, or the risk that remains after all affordable and reasonable control measures have been implemented. Federal government agencies tasked with public safety must be able to effectively inform the public regarding such residual and/or involuntary risks. Failure to properly convey such risk information could be disastrous in many ways, from unsafe public behavior to governmental mistrust.

Such instances, i.e., when risk is imperceptible, not immediately obvious, or hard to comprehend, exemplifies a duality of disseminating scientific work to inform risk behavior. Put simply, scientific work (and its implications) must be understood before it can be effective. At the same time, however, such scientific work may be wrought with uncertainty or difficult concepts to comprehend, even for domain experts. Weighing this duality is a non-trivial issue that scientists must acknowledge before effective risk communication can begin to take place. Additionally, knowing *what* scientific work should be presented is yet another difficult question that scientists must weigh.

Rosener and Russell [67] highlight why risk communication is especially important in the nuclear power industry. In this article, the authors summarize their findings of a task force created to evaluate the emergency planning of California's nuclear power plants. There are two key implications that can be drawn from Rosener and Russell's article. First, when disaster strikes, people turn to the government for help and guidance. Second, unlike natural disasters (e.g., earthquakes, floods, and forest fires) where damages can be readily observed by people, the risks of nuclear power are less discernible; the damage to human life caused by radiation is not straightforward and/or immediate. Thus, in order for the public to take appropriate action during a nuclear disaster, they must be well informed and fully acknowledge the risks thereof. The authors raise many salient questions regarding the quality, quantity, and focus of information required for the public to take unseen nuclear risks seriously. The article by Rosener and Russell further exemplifies the need for clear risk communication in the nuclear industry and helps motivate the guidelines put forth by the NRC.

In recognizing the need for effective risk communication, the NRC has published guidelines [65] that agency employees should abide by when communicating with external stakeholders (e.g., public, interest groups, licensees, activists, and media). Herein I summarize and critique four of the NRCs guidelines for risk communication: 1) know your audience, 2) be realistic about outcomes, 3) tame technical information, and 4) evaluate effectiveness. I focus on these four aspects because they represent guidelines that, based on the risk com-

munication literature, are reasonable, need further clarification, or are virtually impossible to implement. In particular, 1) know your audience and 2) be realistic about outcomes are reasonable guidelines to abide by, 3) tame technical information needs better clarity, and 4) evaluate effectiveness of risk communication is very difficult (if not impossible) to implement. I summarize the NRCs guidelines and then present scholarly literature that defends or refutes the NRC's approach.

In Chapter 3 of the NRC's risk communication guidelines, i.e., "Learning about Your Stakeholders" [65], the authors stress the need for NRC members to learn as much as possible about the audience involved. Specifically, to best prepare for communication, NRC staffers should learn who the people are and what their concerns may be. To learn who the people are, NRC communicators should search the internet, contact other agencies, and/or scour local media for information about the community, including demographics, ethnicities, languages, preponderance of sensitive populations (e.g., pregnant or elderly), local history, media accessibility, local politics, and popular activities. Such knowledge, according to the NRC's guidelines, will help the communicator use appropriate language that will make sense to the stakeholders and better anticipate questions and concerns.

The second portion of Chapter 3, i.e., learning what stakeholder concerns are, is perhaps more useful [65]. In this section, the authors stress that stakeholder concerns can come from a variety of sources including, but not limited to: scientific facts, emotions, misinformation, hidden agendas, local politics, and religious views. In this regard, the NRC's guidelines admit (to some extent) that risk has inherent value judgments and the NRC's communicators should acknowledge this realization. In other words, when anticipating stakeholder concerns, the NRC should look beyond the somewhat obvious scientific, environmental, and/or health topics and include non-scientific aspects such as emotions and politics.

In an article by Fischhoff [68], the author describes several approaches to identifying what facts are worth sharing in risk communication scenarios. This article provides a conceptual framework and commentary that indirectly supports the NRC's "know your audi-

ence" guideline. Fischhoff acknowledges that risk decisions often involve value judgments and other aspects beyond technical analysis. The author then provides a conceptual framework regarding how to determine what information (both technical and non-technical) is needed to make more informed (and thus proper) risk decisions.

Fischhoff suggests that scientists must be involved in the front-end portion of the risk communication process, where decision-relevant information is gleaned from the stakeholders through dialog, open-ended questions, focus groups, and surveys. For example, the author suggests that discussing the risks of prescriptive drugs with a doctor is much more valuable than reading the incomprehensible documents on drug labels because doctors can help weigh the non-technical values inherent in health-related decisions. In sum, Fischhoff concludes that gaining an understanding of the concerns of stakeholders, a guideline put forth by the NRC, is paramount to effective risk decisions and risk communication.

In Chapter 11 of the NRC's guidelines, i.e., "Evaluating the Effectiveness of Risk Communication" [65], the authors note to "be realistic about what effective risk communication looks and feels like". In this chapter, the authors stress that the goal of the NRCs risk communication is to provide clear and accurate information such that all stakeholders are better informed. In other words, the NRC is not trying to reach consensus among all parties involved nor avoid confrontations; instead, the NRC wants the information it provides to be used objectively in the decision making process. Much like "knowing your audience" is perceptive and may improve risk communications, the NRC's policies regarding being realistic about outcomes is quite insightful as well. Specifically, the NRC does not aggrandize itself as an entity that will directly change people's behavior.

The NRC's insights regarding realistic outcomes is supported by an article by Golding, Krimsky, and Plough [69]. In this article, the authors experimented with and evaluated two different ways to convey information about the risks of radon, i.e., technical or narrative. In the technical presentation, the risks of radon were presented in much the same was as the EPA report regarding radon hazards, rewritten to be more readable. The narrative

presentation was written as a dramatized story that focused on a woman trying to better understand radon risks by talking to her college professor neighbor and her personal doctor. The presentations ran as stories in two different newspapers over several days; the technical series ran in the Clinton Daily Item of Clinton, MA, and the narrative series ran in the Fitchburg Sentinel and Enterprise in Fitchburg, MA. Both towns had similar socio-economic demographics and a third city, Worcester City, MA, was used as a control. The authors conducted baseline and post article telephone surveys of 491 total people, with interesting results.

The most telling result is that, although both articles increased knowledge about radon risks (compared to baseline and control), neither article series significantly increased the willingness to test and/or mitigate household radon [69]. Although the sample size was small, these results are quite striking. Even though people gained knowledge that radon exposure caused lung cancer and radon testing was inexpensive, people were still reluctant to change their behavior. In other words, providing pertinent information in easy to digest formats may not actually affect people's behavior. Thus, the NRC's "be realistic about outcomes" is very perceptive, insightful, and accurate.

The NRC's risk communication guidelines contains several chapters regarding messaging and how to communicate technical information. In essence, the NRC guidelines suggest that communicators be accurate, straightforward, consistent, and easy to understand. When dealing with highly scientific content, the NRC recommends the use of analogies and graphics to make the information more understandable. With regard to analogies, the NRC guidelines go so far as to say "CAUTION: indiscriminate use of risk comparisons may be detrimental to your credibility" [65]. In other words, although the NRC guidelines recommend the use of analogies and graphics to convey complex technical material succinctly, it does not offer any insight, background, or basic rules for creating effective analogies or graphics. This gap is a prime example of how the NRC guidelines were well intended but need more specificity to be effective.

What are some "best practices" when visualizing complex yet pertinent information to the non-scientific public during risk assessment scenarios? In a meta-analysis article concerning graphics and risk communication, Lipkus and Hollands [70] identified several key points when using visuals to convey complex information regarding risk. This article evaluated and summarized nearly 100 sources regarding 1) why graphics are effective to communicate risk, 2) what graphics are commonly used to communicate risk, and 3) what are the key issues to ponder when using graphics to communicate risk. There are two take-home points of this article that should be included in the NRC's guidelines. First, the amount of ink used to create graphics should be kept to a minimum; the audience viewing the graphic should do as few mental operations as possible. Second, risk communicators should employ the use of ladder graphics, where risk levels are plotted vertically in relative terms. For example, NRC employees could plot the amount of radiation from a nuclear power plant as a point in relation to the background or naturally occurring radiation. Using such graphics allows participants to see (very quickly) the relative level of harm given the risk. In sum, this article makes specific suggestions regarding how graphics should be used for communicating risk; such specificity is missing from the NRCs guidelines and, as a result, the quality of the NRC's risk communication may suffer.

In Chapter 11 of the NRC's guidelines, the authors advise NRC risk communicators to obtain feedback from stakeholders regarding the effectiveness of their risk communication. To do this suggestion, the NRC recommends several steps, including: reading local newspapers, watching local media outlets, having observers at the meeting, and/or asking the stakeholders for their opinions. Additionally, the NRCs guidelines lists several myths about evaluating risk communication effectiveness, namely, that evaluation is expensive, time consuming, and complicated. Lastly, the NRC says to measure success "based on whether you reach your audience and whether you understand each other's points of view" [65]. Though I commend the NRC's efforts to gauge the effectiveness of its risk communications, the simplicity of the suggested methods greatly underestimates the significant challenges in communicating

risk. In other words, evaluating the actual effectiveness of risk communication in terms of advancing knowledge, changing behavior, or reaching consensus is quite difficult, and falls within the realm of rigorous social science; simply asking audience members if they understood the material is not enough.

Rohrmann [71] wrote an article clarifying how to evaluate the effectiveness of risk communication. In sum, the article highlights the need for well structured and rigorous empirical evaluation of not only the communication, but the entire process of risk communication, including outcomes. To this end, Rohrmann suggests three sets of criteria that must be met to truly evaluate effectiveness. The first is measuring the degree to which the content and communication of risk is valid for a given subject. Here, Rohrmann stresses that, in addition to questioning the comprehensibility and correctness of the information, evaluators should measure how well the messaging stimulates attention. The second set of criteria focuses on the process of risk communication, i.e., making sure relevant parties are involved and that information is exchanged in a two-way communication process. The third set of criteria measures outcomes of risk communications, i.e., the extent of behavioral changes attributed to risk communications.

Rohrmann [71] clearly states that reaching all three criteria requires very careful research designs, including control populations, longitudinal studies, and a good understanding (and measurement) of external influences, e.g., politics, religion, and risk aversion. The author discusses that such research designs require the expertise of highly experienced and talented social researchers, and that quick and dirty research methods are not appropriate for measuring effectiveness in such complex social situations. Rohrmann goes on to say that for most (if not all) risk communications, there is no empirical evaluation at the level required to truly measure effectiveness. In other words, the NRC's guidelines to measure effectiveness are too simple to adequately measure whether or not their risk communications are having positive (or negative) effects on the stakeholder's understanding or behavior.

Although the NRC's guidelines are well intentioned and have great information for scientists to read, there are aspects where the guidelines fall short (particularly in assessing communication effectiveness). The literature in this section was selected to highlight that the NRC, a long-standing, well established, and very experienced government entity that often deals with risk communication, can not fully define the intricacies and complexities of effective risk communication. In other words, the question of *how* scientific work should be communicated to the non-scientific public in a risk scenario does not have clear-cut answers. Of course, there are guidelines that can be followed to maximize chances of effective information transfer (e.g., taming technical information and minimizing ink on graphics), but determining how, why, and what scientific work can influence risk behavior is often beyond the scope of scientists working in such fields.

6.5 Conclusions

I conclude with a discussion of three broad themes that can be drawn from the literature surveyed in this chapter. Such discussion will be framed from the perspective of a scientist and/or engineer working within the field of geohazard monitoring, assessment, and/or mitigation. In particular, my perspective is one of being a researcher or engineer trying to get the scientific work or technological innovation adopted in society.

The first discussion topic is that it is very unlikely that scientific work alone will lead to policy action. If scientists and/or engineers reach a breakthrough in, for example, automated earth dam monitoring, the results alone will not lead to wide-scale, pervasive use of such technologies. This revelation may be difficult for some scientists to comprehend, especially those working in more applied sciences. The idea that breakthrough scientific work will *not* be adopted for reasons *outside of* science may be a hard pill to swallow for scientists in niche disciplines and with years of domain-specific training. In any case, it is important for scientists and engineers to acknowledge that the lack of science and technology adoption is mostly beyond their control or full of complexities; there are many political, economic, and social aspects that play into societal acceptance and use.

The second theme worth discussing (based on the literature surveyed in this chapter) is that, although scientific work may not directly influence people's perceptions of risk, scientists should not simply avoid dissemination altogether. The literature in Section 6.3 shows that the perception of risk is multi-faceted, involving many aspects beyond scientific results (e.g., emotions, trust, and experience); even if there is scientific consensus regarding how and why to avoid particular risks, people may or may not heed such advice. Scientists should not, however, simply "give up" on using scientific evidence to inform risk decisions because, at a minimum, such information may play a subconscious or indirect role in determining people's safety in a risk scenario; *how* the non-scientific public uses scientific evidence to inform decisions comes down to individual choice. What, then, *is* the scientist's responsibility in a risk scenario? Does the scientist's responsibilities end after the scientific evidence has been disseminated? Answers to such questions are context specific. In certain situations, perhaps the scientist has an obligation to communicate in a more persuasive (rather than purely objective) manner. For example, if scientific assessment of an earthen dam suggests that the dam will fail (with high probability) in less than an hour, it seems that scientists have a moral obligation to *persuade* local officials to evacuate the area, rather than simply *inform* people of the likelihood of failure (and the uncertainties involved).

The third and final theme of discussion stems from the previous idea regarding scientific dissemination and the scientist's expectations; "effective" risk communication depends entirely on how "effective" is defined. I argue that "effective" can be defined in one of two ways. To some, "effective" risk communication may pertain to sharing information that changes someone's thoughts, attitudes, and/or behaviors in some way. The other definition of "effective" risk communication could be regarding how "well" the material is presented to the audience, regardless of any substantive outcomes. The differences will determine whether the risk communicator must be *persuasive* or *informative*. For example, a risk communicator following a *persuasive* modality may seek to utilize the audience's emotional responses to incite change. Alternatively, the *informative* communicator may try to avoid emotionality

from the information exchange, focusing instead on the "cold, hard facts". It seems that improving risk communication is difficult, because these two definitions are not necessarily mutually exclusive; scientists are most likely advocates for (or at least have opinions of) the scientific material they are communicating. Though it is nearly impossible for a scientist to be an objective informant sans bias, who else will present the complex scientific material with sufficient expertise? This contradiction may not be rectifiable; it is my opinion that no amount of "further research" will determine the ideal ratio of being *persuasive* versus *informative* for "effective" communication in risk scenarios. In the end, communication is a human endeavor wrought with the complexities of human emotions and behavior; as such, each audience member will have his or her own "ideal" combination required for effective communication to take place.

CHAPTER 7

GENERAL CONCLUSIONS AND FUTURE DIRECTIONS

In this dissertation, we present four technical contributions addressing three research challenges in developing an efficient wireless sensor network (WSN) capable of long-term, autonomous seismic monitoring, i.e., 1) automated seismic event detection, 2) efficient wireless transmission, and 3) precise wireless hardware. We conclude by reviewing how our technical research contributions address these challenges and list the multitude of next steps that we plan to pursue.

7.1 Automated Avalanche Detection

First, in Chapter 2, we present our novel pattern recognition workflow to automatically detect avalanches in passive seismic data. Our contribution, i.e., a novel pattern recognition workflow, addresses the first noted challenge in developing a geohazard monitoring WSN (i.e., automated seismic event detection). Our workflow contributes to the applied machine learning, WSN, and geophysics research communities by providing a proven, step-by-step technique to signal process, data mine, and classify events in passive seismic data. Additionally, our evaluation of different types of machine learning algorithms, majority class reduction techniques, and event selection methods helps inform future scientists and engineers of what algorithms to favor and methods to avoid.

There are many opportunities for future work. First, we plan to continue investigating ways to more intelligently select events in seismic data prior to classification. Second, we intend to experiment with semi-supervised and unsupervised machine learning algorithms to eventually remove the requirement for manual (and tedious) training set identification. Third, we plan to incorporate information from other sensor systems (e.g., multiple geophones, meteorology, and flight path) to create an increasingly robust avalanche detection workflow. Finally, we plan to collaborate with other scientists and engineers to expand this

workflow into other domains that monitor geosystems using seismic data (e.g., earth dam and levee monitoring).

7.2 On-Mote Compressive Sampling

Second, in Chapter 3, we describe and evaluate our novel, lightweight, on-mote compressive sampling (CS) algorithm called Randomized Timing Vector (RTV). Our main contribution, i.e., the RTV algorithm, addresses the second research challenge inherent of geohazard monitoring WSNs (i.e., efficient wireless transmission) by providing a novel, robust, and lightweight on-mote algorithm that reduces transmitted data and, thus, power consumption. Results from our experiments using real hardware show that RTV outperforms two other on-mote CS algorithms in terms of achievable sampling rates, computational overhead, and power consumption. Such work informs WSN scientists and engineers of a novel on-mote compression algorithm that is lightweight and efficient.

For future work, we plan to further evaluate and increase the performance of our RTV algorithm. For example, would it be possible to implement an adaptive RTV algorithm such that a near-optimal compressive sampling rate is set automatically during data acquisition?

7.3 On-Mote Lossy Compression Algorithms

Third, in Chapter 4, we further evaluate our RTV algorithm by rigorously comparing it to four state-of-the-art, on-mote compression techniques in both simulation and hardware. Our contribution further addresses the second research challenged mentioned previously (i.e., efficient wireless transmission) by providing a direct comparison of five on-mote lossy compression algorithms in terms of compression rates, signal recovery errors, classification accuracies (for an event detection task), power consumption, and mote runtimes. Results from our experimentation show that our RTV algorithm performs comparably (and in many cases better than) four state-of-the-art lossy compression algorithms; specially, RTV can guarantee mote power savings without subjugating decompressed signal quality. For scientists and engineers in various research and technology communities, our work shows that compres-

sive sampling "stacks up" against other, perhaps more traditional, on-mote techniques for compressing seismic data.

In terms of future work, we plan to analyze the effects of lossy compression on the scalability of a time-division multiple access enabled WSN. Time division multiple access (TDMA) is one way to implement the scheduling of communication to avoid packet collisions in a wireless sensor network (WSN) [72]. In TDMA, a wireless node is given a specific time slot in each period when that mote can transmit its data. Since RTV provides guaranteed compression rates and, thus, guarantees on the size of data to transmit, RTV could increase the scalability of TDMA WSNs. In other words, less data to transmit equates to a larger sensor network. For example, if a wireless sensor node produces 500 bytes of data per second and transmits at a rate of 7200 bytes per second, then we can have at most 14 nodes in the network to avoid data loss (from queueing); however, if each node produces 250 bytes of compressed data per second, then we can double the size of our network without any queueing errors occurring.

7.4 GeoMoteShield: Custom Hardware for Wireless Geophysics

Fourth, in Chapter 5, we present our custom, low-cost, Arduino-based GeoMoteShield and evaluate its capabilities against other, more expensive seismic acquisition systems. Our contribution addresses the third research challenge of developing a geohazard monitoring WSN (i.e., precise wireless hardware) by providing an inexpensive, validated, Arduino-based prototype that is capable of wireless seismic data acquisition. Results from our lab and field tests show that the GeoMoteShield performs comparably to a much more expensive, traditional *wired* seismograph and "from the ground up" wireless gsMote in terms of precision, accuracy, and time synchronization. Our main contribution to the greater scientific and engineering communities is not only custom hardware that can be used for wireless geophysics, but also field-based evidence showing that Arduino-based systems should *not* be overlooked as a research-grade geophysical sensing tool.

With regard to the GeoMoteShield, there are many avenues of future work. In essence, our future work centers around engineering a robust, field-ready WSN of GeoMoteShields that works as well as (if not better than) a wired seismograph. First, we plan to implement fault tolerance in our WSN; such application-level software will guarantee high data quality in terms of near perfect time synchronization and no missing data packets. Second, we plan to implement dynamic gain settings for the GeoMoteShield; this dynamic parameter setting would allow each mote to adaptively select the optimal gain parameters automatically depending on a history of recently collected samples. Third, we intend to implement a continuous sampling paradigm for long-term monitoring applications; unlike snap-shot data collection, continuous data acquisition contains several implementation challenges involving time-synchronization, collision avoidance, and fault tolerance. Finally, we plan to continue research and development of our GeoMoteShield through controlled tests, domain expert feedback, and iterative engineering.

REFERENCES CITED

[1] B. Tremper. *Staying Alive in Avalanche Terrain*. The Mountaineers Books, Seattle, WA, 2008.

[2] Colorado Avalanche Information Center. Accidents: Statistics. `http://avalanche.state.co.us/accidents/us/`, 2014. Retrieved 07-27-2014.

[3] National Oceanic and Atmospheric Administration. NWS weather fatality, injury and damage statistics. `http://www.nws.noaa.gov/os/hazstats.shtml`, 2013. Retrieved 07-27-2014.

[4] Colorado Department of Transportation. Transportation facts 2010. `http://www.coloradodot.info/library/FactBook/FactBook10-2.pdf/view`, 2010. Retrieved 07-27-2014.

[5] Colorado Department of Transportation. Transportation facts 2011. `http://www.coloradodot.info/library/FactBook/FactBook2011/view`, 2011. Retrieved 07-27-2014.

[6] D. McClung and P. Schaerer. *The Avalanche Handbook*. The Mountaineers Books, Seattle, WA, 2006.

[7] K. Stone, C. Oden, B. Hoenes, D. Hakkarinen, and T. Camp. Hardware for continuous wireless geophysical monitoring. *Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 652–659, 2011.

[8] N. Kimura and S. Latifi. A survey of data compression in wireless sensor networks. *Proceedings of the International Conference on Information Technology: Coding and Computing*, pages 8–13, 2005.

[9] A. Herwijnen and J. Schweizer. Monitoring avalanche activity using a seismic sensor. *Cold Regions Science and Technology*, 69(2-3):165–176, 2011.

[10] B. Biescas, F. Dufour, G. Furdada, G. Khazaradze, and E. Suriñach. Frequency content evolution of snow avalanche seismic signals. *Surveys in Geophysics*, 24:447–464, 2003.

[11] K. Nishimura and K. Izumi. Seismic signals induced by snow avalanche flow. *Natural Hazards*, 15:89–100, 1997.

[12] E. Suriñach, G. Furdada, F. Sabot, B. Biescas, and J. Vilaplana. On the characterization of seismic signals generated by snow avalanches for monitoring purposes. *Annals of Glaciology*, 32(7):268–274, 2001.

[13] B. Leprettre, J.-P. Navarre, and A. Taillefer. First results from a pre-operational system for automatic detection and recognition of seismic signals associated with avalanches. *Journal of Glaciology*, 42(141):352–363, 1996.

[14] B. Leprettre, N. Martin, F. Glangeaud, and J.-P. Navarre. Three-component signal recognition using time, time-frequency, and polarization information– application to seismic detection of avalanches. *IEEE Transactions on Signal Processing*, 46(1):83 – 102, 1998.

[15] B. Leprettre, J.-P. Navarre, J.M. Panel, F. Touvier, A. Taillefer, and J. Roulle. Prototype for operational seismic detection of natural avalanches. *Annals of Glaciology*, 26: 313–318, 1998.

[16] J.-P. Navarre, E. Bourova, J. Roulle, and D. Delio. The seismic detection of avalanches: an information tool for the avalanche forecaster. *Proceedings of the International Snow Science Workshop*, 2009.

[17] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, Boston, MA, 2005.

[18] B. Bessason, G. Eiriksson, O. Thorarinsson, A. Thorarinsson, and S. Einarsson. Automatic detection of avalanches and debris flows by seismic methods. *Journal of Glaciology*, 53(182):461–472, 2007.

[19] Y. Wang, J. Wong, and A. Miner. Anomaly intrusion detection using one class SVM. *Proceedings of the Fifth Annual IEEE SMC Information Assurance Workshop*, 2004.

[20] O. Lartillot and P. Toiviainen. A Matlab toolbox for musical feature extraction from audio. *Proceedings of the 10th International Conference on Digital Audio Effects*, 2007.

[21] A. Klapuri and M. Davy. *Signal Processing Methods for Music Transcription*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[22] A. Orriols and E. Bernadó-Mansilla. The class imbalance problem in learning classifier systems. *Proceedings of the 2005 Workshop on Genetic and Evolutionary Computation*, pages 74–78, 2005.

[23] V. Garcia, J. Sanchez, R. Mollineda, R. Alejo, and A. Sotoca. The class imbalance problem in pattern classification and learning. *Congreso Español de Informática*, pages 283–291, 2007.

[24] S. Yen and Y. Lee. Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications*, 36:5718–5727, 2009.

[25] M. Berthold, N. Cebron, F. Dill, T. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel. KNIME: The Konstanz Information Miner. *Studies in Classification, Data Analysis, and Knowledge Organization*, 2007.

[26] P. Reutemann, B. Pfahringer, and E. Frank. A toolbox for learning from relational data with propositional and multi-instance learners. *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence*, 3339:1017–1023, 2004.

[27] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[28] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak. Compressive wireless sensing. *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 134–142, 2006.

[29] J. Hao, F. Tosato, and R.J. Piechocki. Sequential compressive sensing in wireless sensor networks. *Proceedings of the IEEE Vehicular Technology Conference*, pages 1–5, 2012.

[30] J. Meng, H. Li, and Z. Han. Sparse event detection in wireless sensor networks using compressive sensing. *Proceedings of the Conference of Information Sciences and Systems*, pages 181–185, 2009.

[31] C. Luo, F. Wu, J. Sun, and C. Chen. Compressive data gathering for large-scale wireless sensor networks. *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 145–156, 2009.

[32] Z. Charbiwala, Y. Kim, S. Zahedi, J. Friedman, and M. Srivastava. Energy efficient sampling for event detection in wireless sensor networks. *Proceedings of the 14th ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, pages 419–424, 2009.

[33] Z. Charbiwala, Y. Kim, S. Zahedi, R. Balani, and M. Srivastava. Weighted ℓ_1 minimization for event detection in sensor networks. Technical report, University of California, Los Angeles, 2009.

[34] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst. Compressed sensing for real-time energy-efficient ECG compression on wireless body sensor nodes. *IEEE Transactions on Biomedical Engineering*, 58(9):2456–2466, 2011.

[35] R. Durstenfeld. Algorithm 235: Random permutation. *Communications of the ACM*, 7(7):420, 1964.

[36] E. Candes and J. Romberg. ℓ_1-magic Matlab library. `http://users.ece.gatech.edu/~justin/l1magic/`, October 2005. Retrieved 07-27-2014.

[37] E. Capo-Chichi, H. Guyennet, and J. Friedt. K-RLE: A new data compression algorithm for wireless sensor network. *Proceedings of the IEEE International Conference on Sensor Technologies and Applications*, pages 502–507, 2009.

[38] T. Schoellhammer, E. Osterweil, B. Greenstein, M. Wimbrow, and D. Estrin. Lightweight temporal compression of microclimate datasets. *Proceedings of the IEEE International Conference on Local Computer Networks*, pages 516–524, 2004.

[39] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*, pages 13–24, 2004.

[40] S. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1997.

[41] M. Rubin and T. Camp. On-mote compressive sampling to reduce power consumption for wireless sensors. *Proceedings of the IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2013.

[42] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, and K. Araki. Practical data compression in wireless sensor networks: A survey. *Journal of Network and Computer Applications*, 35:37–59, 2012.

[43] Open Music Labs. Fast Fourier transform library. `http://wiki.openmusiclabs.com/wiki/ArduinoFFT`, 2014. Retrieved 07-27-2014.

[44] E. Candes, M. Wakin, and S. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5):877–905, 2008.

[45] M. Rubin, T. Camp, A. Herwijnen, and J. Schweizer. Automatically detecting avalanche events in passive seismic data. *Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1:13–20, 2012.

[46] Memsic. TelosB Datasheet. Technical report, Memsic Inc., 2010.

[47] S. Kobayashi, M. Banzi, D. Cuartielles, T. Igoe, G. Martino, and D. Mellis. Arduino Fio Wireless Mote. `http://arduino.cc/en/Main/ArduinoBoardFio`, 2014. Retrieved 06-29-2014.

[48] Geometrics. Geometrics geode datasheet. http://www.geometrics.com/geometrics-products/seismographs/geode/specifications/, 2014. Retrieved 06-29-2014.

[49] Analog Devices. Analog devices 7714 datasheet. http://www.analog.com/static/imported-files/data_sheets/AD7714.pdf, 1998. Retrieved 06-29-2014.

[50] B. Latour. *Science in Action*. Harvard University Press, 1987.

[51] D. Sarewitz. *Frontiers of Illusion*. Temple University Press, 1996.

[52] N. Oreskes and E. Conway. *Merchants of Doubt*. Bloomsbury Press, 2010.

[53] M. Downton, R. Morss, O. Wilhelmi, E. Gruntfest, and M. Higgins. Interactions between scientific uncertainty and flood management decisions: Two case studies in Colorado. *Environmental Hazards*, 6:134–146, 2005.

[54] A. Meltsner. The communication of scientific information to the wider public: The case of seismology in California. *Minerva*, 17(3):331–354, 1979.

[55] A. Marcus. Risk, uncertainty, and scientific judgement. *Minerva*, 26(1):138–152, 1988.

[56] W. Ascher. Scientific information and uncertainty: Challenges for the use of science in policymaking. *Science and Engineering Ethics*, 10:437–455, 2004.

[57] C. Weiss. Scientific uncertainty and science-based precaution. *International Environmental Agreements: Politics, Law and Economics*, 3:137–166, 2003.

[58] A. Sol and H. Turan. The Ethics of Earthquake Prediction. *Science and Engineering Ethics*, 10:655–666, 2004.

[59] N. Nosengo. Italian court finds seismologists guilty of manslaughter. *Nature News*, 2012.

[60] S. Hall. Scientists on trial: At fault? *Nature*, 477:264–269, 2011.

[61] R. Johnson and M. Scicchitano. Uncertainty, risk, trust, and information: Public perceptions of environmental issues and willingness to take action. *Policy Studies Journal*, 28(3):633–647, 2000.

[62] W. Kellens, R. Zaalberg, T. Neutens, W. Vanneuville, and P. De Maeyer. An Analysis of the Public Perception of Flood Risk on the Belgian Coast. *Risk Analysis*, 31(7): 1055–1068, 2011.

[63] R. Lidskog. In science we trust? on the relation between scientific knowledge, risk consciousness and public trust. *ACTA Sociologica*, 39(1):31–56, 1996.

[64] D. Liverman. Communicating geological hazards: Educating, training and assisting geoscientists in communication skills. *Geophysical Hazards*, pages 41–55, 2010.

[65] Nuclear Regulatory Commission. Effective risk communication: The nuclear regulatory commission's guidelines for external risk communication. `http://pbadupws.nrc.gov/docs/ML0406/ML040690412.pdf`, 2014. Retrieved 07-27-2014.

[66] L. Thomas. Risk communication: Why we must talk about risk. *Environment*, 28(4-5): 40, 1986.

[67] J. Rosener and S. Russell. Cows, sirens, iodine, and public education about the risks of nuclear power plants. *Science, Technology, and Human Values*, 12(3-4):111–115, 1987.

[68] B. Fischhoff. Communicating unto others.... *Reliability Engineering and Safety System*, 59:63–72, 1998.

[69] D. Golding, S. Krimsky, and A. Plough. Evaluating risk communication: Narrative vs. technical presentations of information about radon. *Risk Analysis*, 12:27–35, 1992.

[70] I. Lipkus and J. Hollands. The visual communicatino of risk. *Journal of the National Cancer Institute Monographs*, 25:149–163, 1999.

[71] R. Rohrmann. The evaluation of risk communication effectiveness. *Acta Psychologica*, 82:169–192, 1992.

[72] S. Ergen and P. Varaiya. TDMA scheduling algorithms for wireless sensor networks. *Wireless Networks*, 16(4):985–997, 2010.

[73] E. Candes. Compressive sampling. *International Congress of Mathematicians*, 3:1433–1452, 2006.

[74] E. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

[75] D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4): 1289–1306, 2006.

[76] E. Candes and M. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, March 2008.

[77] R. Baraniuk. Compressive sensing. *IEEE Signal Processing Magazine*, 24(4):118–121, July 2007.

[78] D. Mackenzie. Compressed sensing makes every pixel count. *What's Happening in the Mathematical Science, American Mathematical Society*, 7:114–127, 2009.

[79] M. Davenport, M. Duarte, Y. Eldar, and G. Kutyniok. *Introduction to Compressed Sensing: Chapter in Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012.

[80] M. Fornasier and H. Rauhut. *Compressive Sensing: Chapter in Part 2 of the Handbook of Mathematical Methods in Imaging*. Springer, 2011.

[81] J. Tropp and A. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.

APPENDIX A - COMPRESSIVE SAMPLING

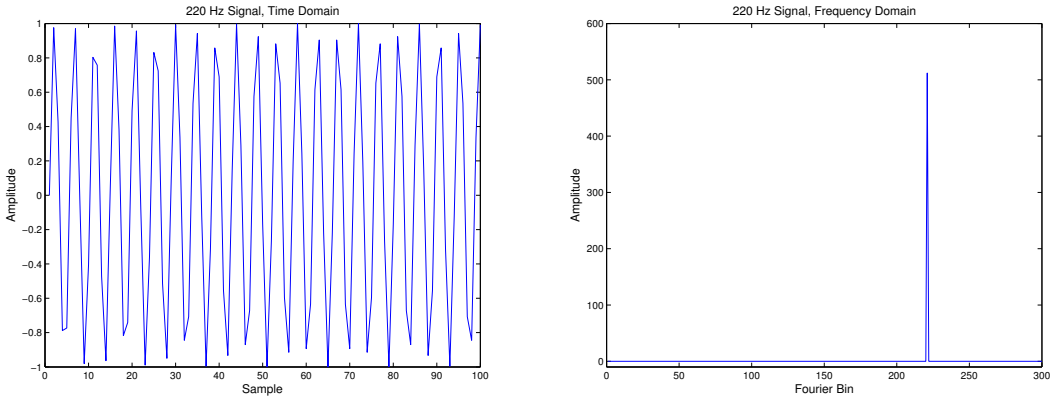
In this appendix, we summarize the technical mechanics of compressive sampling in more detail. A new and exciting field known as "Compressive Sensing" or "Compressive Sampling" has been growing over the past decade. Compressive sensing replaces the traditional (and often wasteful) notion of "sample THEN compress" with "compress WHILE sampling." For clarification, this section is presented as an amalgamation of many publications from journal articles [73–78] and book chapters [79, 80]. It is our goal to present a high-level and intuitive summary of what compressive sensing is and how it works. For more in-depth explanations regarding the foundations of compressive sensing (including mathematical proofs of *why* it works), motivated readers are encouraged to read the literature listed in the bibliography (especially [73, 75, 76]) and noted on Rice University's online repository (`http://dsp.rice.edu/cs`).

Compressive sensing offers solutions to the following observations posed by D. Donoho in [75]: "Why go to so much effort to acquire all the data when most of what we get will be thrown away?" and "Can't we just directly measure the part that won't end up being thrown away?". As written by Candes and Wakin in [44]: "compressive sensing theory asserts that one can recover certain signals and images from far fewer samples than traditionally thought." In essence, compressive sensing uses numerical optimization techniques to recover full-length signals from only a small number of collected samples. Compression (i.e., data reduction) occurs WHILE sensing and not afterwards (hence compressive *sensing* or *sampling*).

This section is organized as follows. First, we discuss an important property of the original signal: sparsity. Next, we describe how compression is achieved via an inner product of the signal with a measurement matrix. Lastly, we summarize how the original signal can be recovered from the compressed version, including several types of algorithms used.

A.1 Sparsity

As noted by Mackenzie in [78], the information content of an image (or signal) will determine how well compressive sensing will work, i.e., the achievable resolution of the recovered image. More specifically, images with low information can be recovered almost perfectly, while images with high information (e.g., white noise with no discernible pattern) cannot be reconstructed. Signals with low information can be represented or conveyed with only a few number of descriptors or patterns. Sparsity is simply another word for low information; sparse signals contain mostly zeros, and the majority of the signal's content can be expressed in just a few values or transform coefficients. Many real-world signals (e.g., images) are sparse with regard to some transform basis (e.g., Fourier, wavelet, or Gabor). For example, a 220 Hz sine wave is not sparse in the time domain (e.g., Figure A.1(a)), yet the signal is sparse in the frequency domain (e.g., Figure A.1(b)).



(a) time domain (b) frequency domain

Figure A.1: A 220 Hz sine wave, plotted in the (a) time domain and the (b) frequency domain.

More formally, the signal x can be represented as the product of the transform matrix Ψ (e.g., Fourier bins) and their associated coefficients α (e.g., power), where:

$$x = \Psi\alpha$$

The signal x is sparse in some domain Ψ when most of the coefficients α are zero. Given that the signal x is sparse in some domain Ψ, few measurements of the original signal are needed for compressive sensing to work. Briefly, a signal is k-sparse when it can be represented by at most k non-zero coefficients.

A.2 Measurement Matrices

Compressive sensing achieves data reduction by transforming the original signal x of length N to an observation vector y of length M via a simple multiplication by an $M \times N$ measurement matrix Φ. An important property of this measurement matrix Φ is that it must be maximally incoherent to the sparsity domain Ψ. Incoherence between Φ and Ψ means that there should be minimal correlation between any two pairs of elements; thus, the measurement matrix should not match the signal structure. Put another way, if Φ and Ψ contain elements that are highly correlated, than the matrices approach increasing coherency.

The measurement matrix Φ must also abide by: 1) the null-space property (NSP) and 2) the restricted isometry property (RIP). At the highest level, these properties basically state that the measurement matrix should not "zero-out" any of the original signal (NSP) and the columns should be orthogonal (RIP). There are several types of $M \times N$ measurement matrices that abide by the NSP and RIP principles, e.g., random Gaussian, Bernoulli, random Fourier, noiselets, and random binary [73, 80]. It has been shown that an independent and identically distributed (i.i.d.) random Gaussian $M \times N$ measurement matrix will abide by RIP with an overwhelmingly high probability [77]. Thus, one can create an effective $M \times N$ measurement matrix Φ by simply creating an i.i.d. random matrix with Gaussian distribution (e.g., in Matlab this would be phi = randn(M,N)).

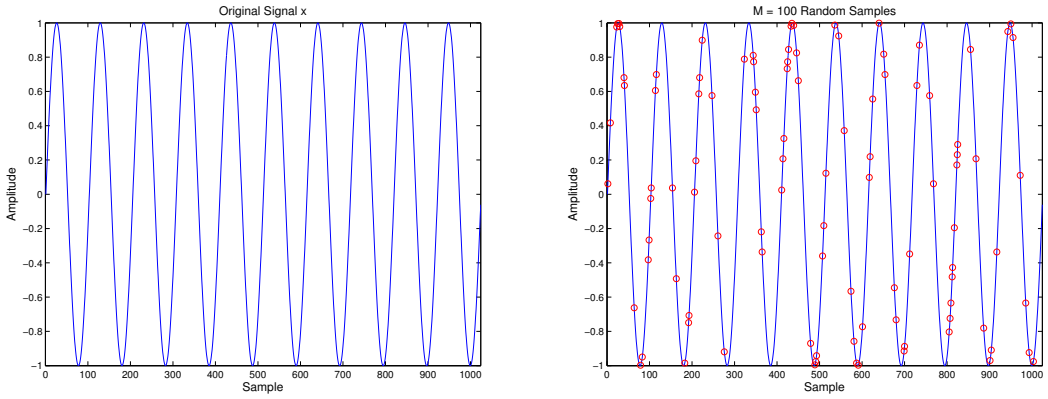
Once we have established our $M \times N$ measurement matrix Φ (e.g., random Gaussian or binary), we can apply this matrix to the original signal x using a simple matrix-vector multiplication to obtain the observation vector y of length M:

$$y = \Phi x$$

For example, if our measurement matrix Φ was a random binary matrix, such as

$$\Phi_{binary} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix},$$

then our observation vector y would consist of random samples from our original signal x (e.g., Figure A.2). Using this measurement matrix Φ_{binary}, we are effectively compressing WHILE sensing; in other words, instead of saving all N samples from signal x, we are only saving M samples in our observation vector y. In other words, y can be acquired directly in compressed form; thus, there is no need to sample and store the entire signal x prior to compression.



(a) original signal (b) M random samples

Figure A.2: In compressive sensing, data reduction of (a) a full signal can be achieved by (b) collecting M random samples from the original signal.

A.3 Signal Recovery

In a real world scenario that has compressed while sensing, we are given (1) the sensed (compressed) observation vector y of length M and (2) the $M \times N$ measurement matrix Φ used to project the original signal x onto y. Given this information (i.e., y and Φ), we then try to recover the original signal x by solving the following underdetermined linear system:

$$y = \Phi x.$$

To solve this underdetermined system, we employ numerical optimization methods to approximate the original signal. More commonly, we use ℓ_1-norm minimization to find the approximate signal $\hat{x}$ that minimizes the error in the linear system in $y = \Phi\hat{x}$. Mathematically:

$$min||\hat{x}||_{\ell_1}, \quad \text{subject to} \quad y = \Phi\hat{x}.$$

When the signal is sufficiently sparse, then ℓ_1-norm minimization will exactly recover the original signal x. In addition, ℓ_1-norm minimization can be solved very efficiently with linear programming (e.g., Basis Pursuit). Basically, we try to find the approximate signal $\hat{x}$ that has the minimum ℓ_1-norm given $y = \Phi x$. It is important to mention that ℓ_1-norm minimization outperforms both ℓ_2-norm and ℓ_0-norm minimization; that is, the ℓ_2 norm approach will rarely find a k-sparse solution and the ℓ_0-norm is NP-Hard [77].

APPENDIX B - COMPRESSIVE SAMPLING ON SEISMIC DATA

In this appendix, we show simulation results from testing six combinations of compressive sampling (CS) recovery algorithms and assumed sparsity domains on a real-world seismic data set. In particular, we compared ℓ_1-norm minimization (L1) [76], reweighted ℓ_1-norm minimization (RWL1) [44], and orthogonal matching pursuit (OMP) [81] recovery algorithms while assuming sparsity in either the frequency (Fourier) or time-frequency (Gabor atoms) domain. Results from this work shows that RWL1, assuming sparsity in the time-frequency domain, has superior performance to the other five combinations tested for recovering passive seismic data.

B.1 Recovery Algorithms

As mentioned, we tested three different CS recovery algorithms: ℓ_1-norm minimization (L1), reweighted ℓ_1-norm minimization (RWL1), and orthogonal matching pursuit (OMP). There are many other types of signal recovery algorithms used in compressive sensing (e.g., Total Variation, Iterative-Hard Thresholding, and Matching Pursuit [80]) that work well depending on the dimensionality and sparsity domains. We chose L1, RWL1, and OMP for the following reasons.

First, L1 seems to be the most common compressive sensing recovery algorithm and is, thus, the starting point in our experimentation. As mentioned in Appendix A, L1 can be solved efficiently using linear programming (e.g., Basis Pursuit). In other words, we chose to experiment with L1 because it provides a base line from which to build.

Second, RWL1 is an iterative algorithm that attempts to improve L1 by introducing weighting into the ℓ_1 minimization problem [44]. More specifically, weighting helps increase the sensitivity to the small (but not insignificant) non-zero coefficients by "penalizing" large coefficients. RWL1 works as follows. To initialize, all N elements in weight vector W are set

to one (where N is the length of the signal x). Next, solve the weighted ℓ_1-norm minimization problem (where t represents the iteration number and W is the weight vector):

$$x^t = argmin||W^t x||_{\ell_1}, \quad \text{subject to} \quad y = \Phi x.$$

Next, for each $i = \{1, 2, \ldots, N\}$, we update the weight vector W for the next iteration (i.e., $t + 1$) according to the equation:

$$w_i^{t+1} = \frac{1}{|x_i^t| + \epsilon} \quad ,$$

where ϵ is a user specified parameter. The algorithm terminates when t reaches a predetermined maximum number of iterations. In this work, we chose to experiment with RWL1 as one of our recovery algorithms because RWL1 helps increase the sensitivity to the smaller, non-dominating, non-zero sparse coefficients of the original signal [44].

Lastly, OMP is a greedy algorithm that attempts to find the approximate signal $\hat{x}$ in an iterative manner by systematically finding the columns of Φ (the compressive sensing measurement matrix) that most contribute to the observations y (in terms of ℓ_2-norm) [79, 81]. (See Appendix A for more information regarding the mechanics of compressive sensing.) OMP works as follows: for each iteration, the algorithm finds the column of the measurement matrix Φ that is most correlated with the remaining observation vector y. This contribution is subtracted from the observation vector y before the next iteration. At each iteration, the recovered signal $\hat{x}$ is estimated using a least squared method according to the following system:

$$\hat{x_t} = argmin_{\hat{x}}||y - \Phi_t \hat{x}||_{\ell_2},$$

where t represents the iteration number, $\hat{x}$ is the approximate signal, Φ_t is the measurement matrix, and y is the remaining observation vector [81]. Initially, Φ_0 is as an empty matrix, which gets filled in with each iteration (i.e., Φ_t). The algorithm terminates after k iterations (for k-sparse signals), revealing an estimated signal $\hat{x}$. In our experiments, we selected to use OMP because it represents a "quick and dirty" algorithm with fast runtimes for full signal recovery.

B.2 Simulation Results

Our simulation experiments were tested on real-world seismic data containing avalanches (see Chapter 2 for details). Specifically, we simulated CS on 33 five-minute chunks of data, each containing a slab avalanche event that lasted, on average, 24.2 seconds in length. We simulated nine different rates of CS by using random binary measurement matrices of different sizes. A random binary measurement matrix, which contains one randomly placed '1' per row and at most one '1' per column (zeros elsewhere) is advantageous because compression can be achieved without performing costly matrix multiplications (see Chapter 3 for details). In other words, the $M \times N$ measurement matrices used to compress the signal went from $M = 0.1N$ to $M = 0.9N$, where M is the length of the compressed signal and N is the length of the original signal. CS compression using a random binary measurement matrix is equivalent to randomly sampling M out of N samples in time.

Our simulation workflow consisted of performing CS on chunks of $N = 512$ consecutive data points of the original signal. The full-length recovered signal was built by concatenating each $\widehat{x}$ recovered signal (of length $N = 512$). Figure B.1 shows our workflow; in Figure B.1, y is the compressed signal encoding, Φ is a random binary measurement matrix, Ψ is the sparsity domain transform function, and $\hat{\alpha}$ are the estimated sparsity domain coefficients.

After compressively sampling the signal, we tested six combinations of recovery algorithms and assumed sparsity domains. Figure B.2 shows two frequency spectrograms: an original slab avalanche with 100% sampling and a recovered slab avalanche from 30% compressive sampling. In both figures, the slab avalanche occurred at time 150 seconds and had significant low frequency energy.

After simulating CS compression, we analyzed the recovered signals in terms of compression rate, signal error, and avalanche event classification accuracy. See Section 4.1.1 for details regarding how compression rate is calculated. For the signal error, we calculated the normalized root mean square error (NRMSE) of each recovered signal versus the original signal. See Section 3.3.1 for details on how NRMSE is calculated. In terms of classification

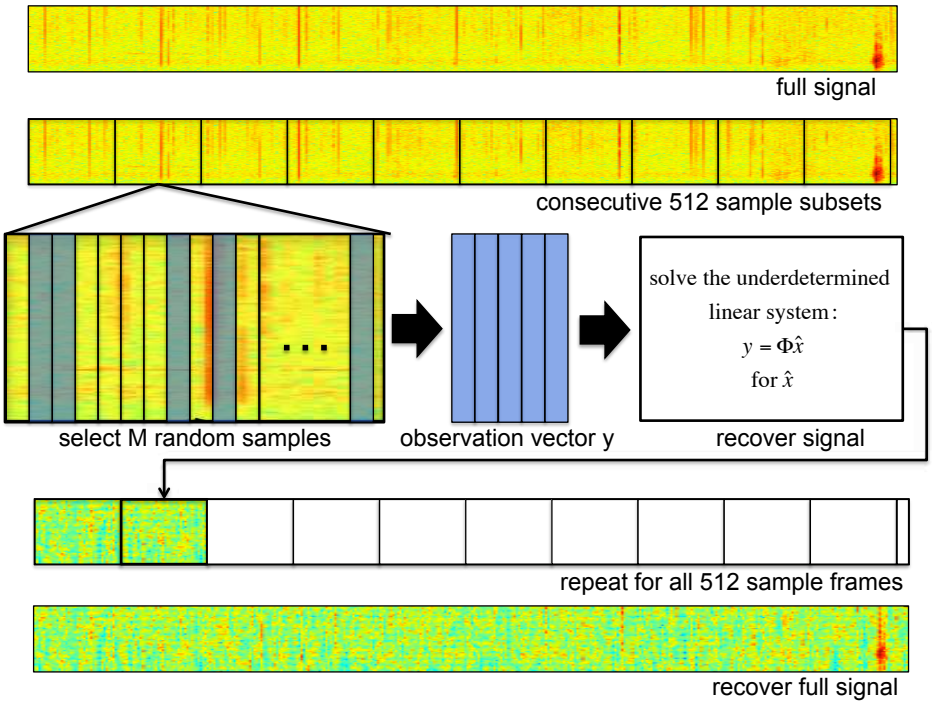
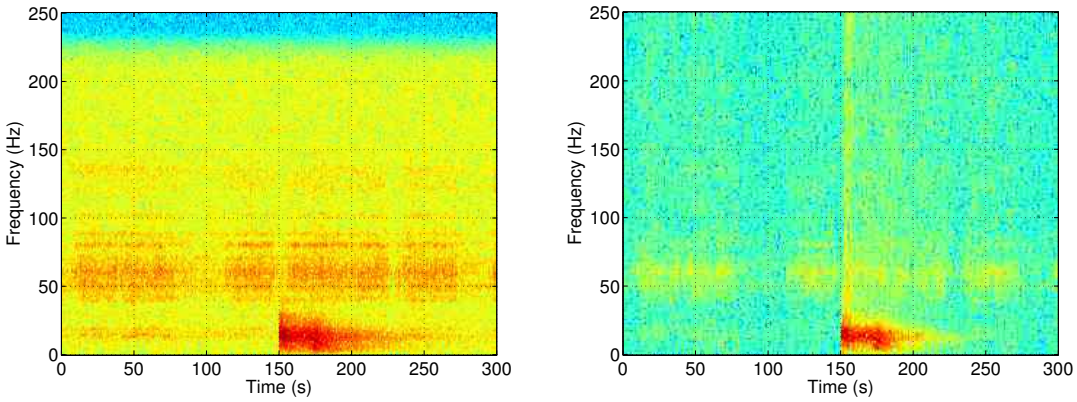


Figure B.1: The simulation workflow used to test compressive sampling.



(a) 100% sampling

(b) 30% compressive sampling, RWL1 recovery from time-frequency domain

Figure B.2: (a) A spectrogram of the original signal of slab #13 with 100% full sampling. (b) A spectrogram of the recovered signal from 30% compressive sampling (70% compression ratio) via RWL1 recovery assuming sparsity in the time-frequency domain.

accuracy, we ran the original and recovered data through the avalanche detection workflow described in Chapter 2. Our results show that the RWL1 recovery algorithm assuming

sparsity in the time-frequency domain performed best; RWL1-Gabor had the lowest reconstruction errors (Figure B.3) and highest classification accuracies overall (Figure B.4). For example, with signals recovered from 30% of the data (70% compression), we were able to obtain 90.7% classification accuracy (compared to 92.4% accuracy with full sampling or 0% compression).

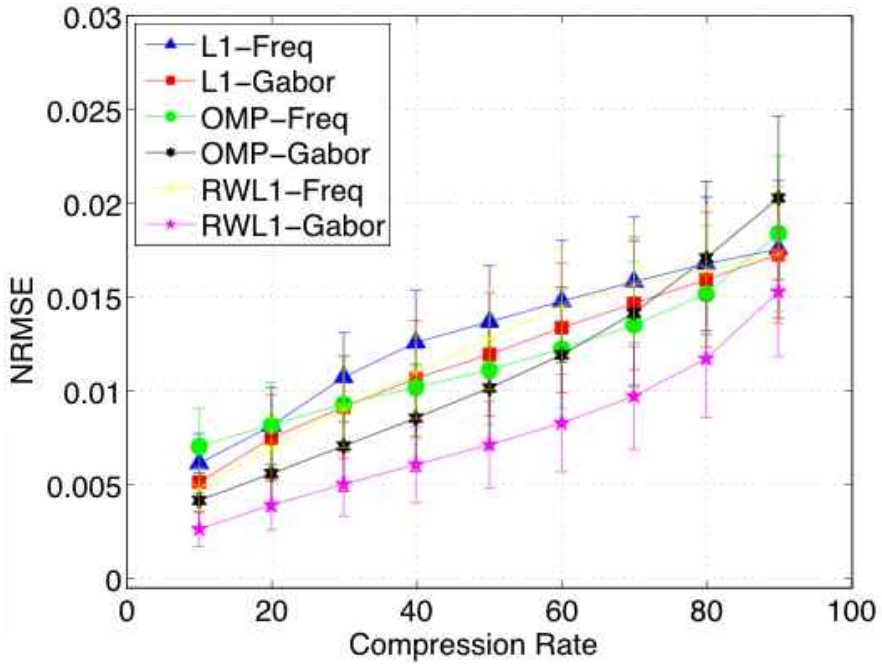


Figure B.3: The mean NRMSE with 95% confidence intervals for the six combinations of CS tested on real-world seismic data containing avalanches.

The presented results demonstrate that RWL1 recovery, assuming sparsity in the time-frequency domain (Gabor atoms), is the best performing recovery method tested. We hypothesize that RWL1-Gabor works best for two reasons. First, the avalanche signal appears most perceptibly sparse in the time-frequency domain; in other words, both time *and* frequency information is needed to determine an avalanche event. Second, RWL1 helps illustrate subtle components of the recovered signal, which may help with differentiating avalanches from non-avalanche signals.

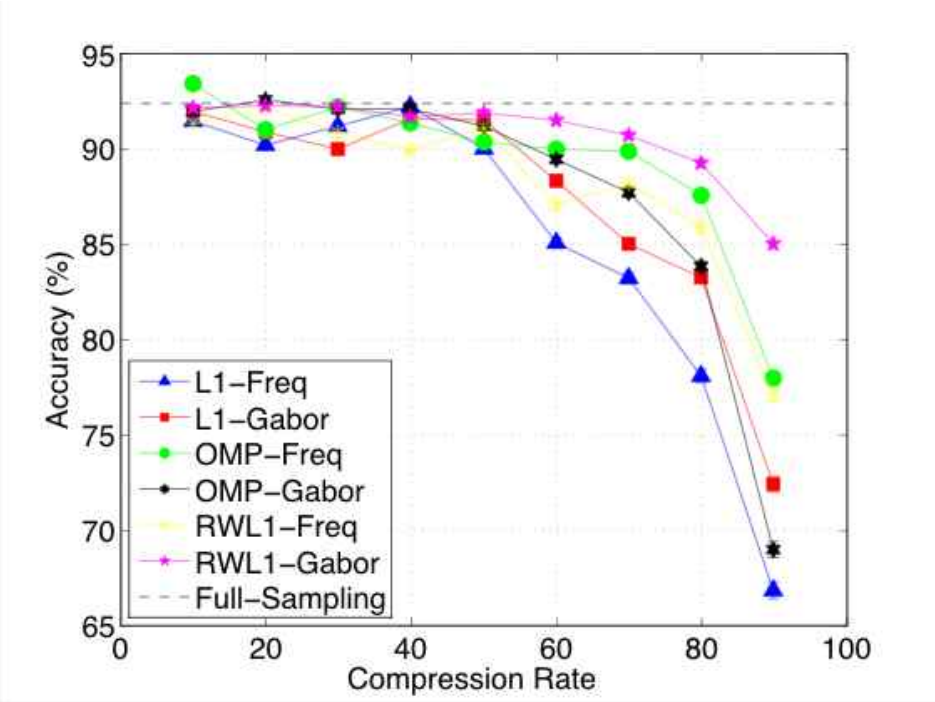


Figure B.4: The mean classification accuracy with 95% confidence intervals for the six combinations of CS tested on real-world seismic data containing avalanches.