THESIS

UNATTENDED ACOUSTIC SENSOR SYSTEMS FOR NOISE MONITORING IN NATIONAL PARKS

Submitted by

Vladimir Yaremenko

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2017

ABSTRACT

UNATTENDED ACOUSTIC SENSOR SYSTEMS FOR NOISE MONITORING IN NATIONAL PARKS

Detection and classification of transient acoustic signals is a difficult problem. The problem is often complicated by factors such as the variety of sources that may be encountered, the presence of strong interference and substantial variations in the acoustic environment. Furthermore, for most applications of transient detection and classification, such as speech recognition and environmental monitoring, online detection and classification of these transient events is required. This is even more crucial for applications such as environmental monitoring as it is often done at remote locations where it is unfeasible to set up a large, general-purpose processing system. Instead, some type of custom-designed system is needed which is power efficient yet able to run the necessary signal processing algorithms in near real-time.

In this thesis, we describe a custom-designed environmental monitoring system (EMS) which was specifically designed for monitoring air traffic and other sources of interest in national parks. More specifically, this thesis focuses on the capabilities of the EMS and how transient detection, classification and tracking are implemented on it.

The Sparse Coefficient State Tracking (SCST) transient detection and classification algorithm was implemented on the EMS board in order to detect and classify transient events. This algorithm was chosen because it was designed for this particular application and was shown to have superior performance compared to other algorithms commonly used for transient detection and classification. The SCST algorithm was implemented on an Artix 7 FPGA with parts of the algorithm running as dedicated custom logic and other parts running sequentially on a soft-core processor. In this thesis, the partitioning and pipelining of this algorithm is explained. Each of the partitions was tested independently to very their functionality with respect to the overall system. Furthermore, the entire SCST algorithm was tested in the field on actual acoustic data and the performance of this implementation was evaluated using receiver operator characteristic (ROC) curves and confusion matrices. In this test the FPGA implementation of SCST was able to achieve acceptable source detection and classification results despite a difficult data set and limited training data.

The tracking of acoustic sources is done through successive direction of arrival (DOA) angle estimation using a wideband extension of the Capon beamforming algorithm. This algorithm was also implemented on the EMS in order to provide real-time DOA estimates for the detected sources. This algorithm was partitioned into several stages with some stages implemented in custom logic while others were implemented as software running on the soft-core processor. Just as with SCST, each partition of this beamforming algorithm was verified independently and then a full system test was conducted to evaluate whether it would be able to track an airborne source. For the full system test, a model airplane was flown at various trajectories relative to the EMS and the trajectories estimated by the system were compared to the ground truth. Although in this test the accuracy of the DOA estimates could not be evaluated, it was show that the algorithm was able to approximately form the general trajectory of a moving source which is sufficient for our application as only a general heading of the acoustic sources is desired.

ACKNOWLEDGEMENTS

I would first like to thank my advisor, Dr. Mahmood Azimi-Sadjadi, who has looked over many versions of this document and helped me make it something I can be proud of. He has taught me a lot about signal processing and academic writing over the years and I am grateful to have had him as my advisor.

I would like to thank my committee members, Dr. Ali Pezeshki and Dr. Chuck Anderson, for their time and assistance during the course of my research.

I would like to thank the National Park Service (NPS) for providing the funding used in this project under contract number P14AC01166. I would also like to thank Dr. Kurt Fristrup for his help in shaping the direction of the entire EMS project (of which only a part is covered in this document) and his continued support of this project.

I would like to thank my colleagues in the Signal and Image Processing Lab. A special thanks to Jarrod Zacher for his help with the field tests of the EMS system, constructing various pieces of hardware for the system and many other things without which this project would never be where it is today. Also, a special thanks to Jack Hall for proofreading this document and offering great commentary and suggestions that greatly improved this work.

I would also like to thank Iman Babazadeh and Ashley Pipkin for their help in conducting the fields tests presented in this document. They have spent a great deal of time helping me with these tests and I am truly grateful.

Finally, I would like to thank my family for their unwavering support and guidance throughout my entire education.

DEDICATION

To my parents, Olga and Alexander.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Problem Statement

Transient acoustic signal detection and classification has many valuable applications including speech recognition [1–3], surveillance [4], and noise monitoring [5]. The problem can be complicated by many factors including the large variety of sources that may be encountered, the variations in acoustic environments in which these sources are present and presence of strong interference signals that mask the sources of interest.

The work in this thesis is motivated by a cooperative agreement with the National Park Service (NPS) which is interested in transient acoustic signal detection and classification for the purpose of monitoring noise levels in national parks. More specifically, the goals of this work are to detect, classify, and track man-made airborne sources such as commercial and military planes and helicopters in order to identify areas of heavy noise pollution in the parks and study the effects this might have on the park's ecosystems. In the sequel, we will refer to any acoustic signal of interest as a *source* while any other acoustic signal which we do not care about but might be present in the observed data will be labeled as *interference*. The word *signal*, on the other hand, will be used to refer to any acoustic signature including sources and interference, while *noise* will be used to refer to ambient background noise.

Currently, this analysis is done manually by experts on data gathered by environmental monitoring stations set up around some parks. However, this process is very labor intensive as it requires technicians to sort through hundreds of hours of acoustic data to manually label and identify events of interest. As a result, it is impractical to deploy a large number of stations for finer spatial resolution monitoring.

The focus of this work is on the development of hardware, firmware, and software for an acoustic monitoring system which will be able to automatically detect, classify and track the sources of interest with very minimal human involvement. These systems will be deployed in the parks for extended periods of time (e.g. several months) without requiring to store a large amount of acoustic data and then post-process the collected data to identify sources of interest. Thus, these systems will need to be able to autonomously detect, classify, and report the type and number of sources of interest in near real-time from streaming acoustic data. The algorithms that will be implemented on these systems will need to be able to detect the start and the end of a source signal without having access to the entire data sequence in which the source was present at any given instant in time. Once the source is detected, the system will need to identify the source type (out of a pre-defined set) and assign an appropriate class label to the detected source. Furthermore, the system will need to generate approximate trajectories of the moving sources for their entire duration. These trajectories will be able to help NPS combine the detection and classification data from different EMS systems deployed in neighboring locations and get accurate estimates of the actual number of aircraft flying over the park as well as their flight paths.

1.2 Literature Review - Transient Detection and Classification

As mentioned previously, there are many challenges to detection and classification of transient acoustic signals. Due to the reasons stated earlier, there are only a few methods which could work for detecting and classifying man-made sources in national parks. Commonly, transient detection is done by having a model for observations when no transient signals are present (null hypothesis) and then looking for changes in the observed data which do not fit this model [6–8]. Once a transient signal is detected, Hidden Markov Model (HMM)-based classification schemes are often used to classify them [1, 9–11] as these models can deal with significant data variability and easily model complex dependencies between consecutive observation vectors which are often found in many transient signals. For instance in [9], the authors break up a speech signal into overlapping 20 ms frames and perform an 8th order Linear Predictive Coding (LPC) analysis on each frame to create a 12-dimensional Cepstral vector representing each frame. These vectors are then used as inputs into discrete HMMs, which were trained on similar data, to determine the one that best fits this particular test sequence. In [11], the authors use a similar approach to classify objects from sonar returns. They formed feature vectors from overlapping segments of the data and used these vectors to train and test HMM models. The authors tested various techniques for extracting feature vectors from these frames including performing short time Fourier Transform (STFT) on the frames, fitting an auto regressive model to the observations in the frames and using the coefficients of the model as the feature vector, as well as using wavelet coefficients obtained by taking the wavelet packet transform of the frames. The class labels were decided based on which HMM model produced the highest likelihood ratio when applied to the test data.

The problem with the HMM-based classification methods is that they are very susceptible to structured interference as they assume only one of the transient signals modeled by the HMMs is present in the data. This makes them almost useless in this type of application where strong interference (e.g., rain, thunder, wind) is often present. In [12], the authors attempted to remedy this problem by modeling the background interference as an HMM that is always present and the sources as HMMs that may or may not be present. The authors propose 2 ways to detect the presence of the new signal. The first method [12] combines the two (or more) superimposed transients into one by taking the Kronecker product of their state transition matrices. Then, the authors use a CUSUM (cumulative sum) procedure based on Page's test [13] in order to find the start and end times of the superimposed HMM signal. Once the log likelihood ratio of multiple superimposed signals being present vs only one signal being present as computed in this CUSUM fashion exceeds a pre-defined threshold then a detection is declared. While straight forward, this method becomes computationally intractable for HMM signal models with large numbers of states or multiple superimposed signals. The second proposed method [12] assumes independence between the state sequences of the superimposed signals given the observation sequence. This method also allows for the detection of superimposed signals but is more computationally efficient than the first method allowing for multiple and/or more complex superimposed signals to be detected without numerical issues. As with the first method, a CUSUM procedure is used with these independent signal models in order to detect the start and end of a superimposed HMM signal. While these methods do address the issue of interference being present during a source event, the proposed detection procedure assumes that there is some signal (source or interference modeled by an HMM) always present in the data. Again, this is not the case in our application as wind and other natural interference events occur sporadically and there are often quiet periods in the data where no source or interference is present. Furthermore, it requires the computation of a log likelihood ratio for each possible combination of source and interference(s) for each observation which can become computationally expensive if a lot of interference signals are considered which may be the case in our application depending on the location which is being monitored.

There are also some methods that deal with transient detection and classification simultaneously. For example, the authors in [14] proposed using a wavelet network to detect and classify transient sources. The wavelet network is a combination of a wavelet transform followed by an artificial neural network with each wavelet coefficient being an input to the neural network. The authors then split up the incoming data stream into consecutive data segments of the same size and apply these segments to the wavelet network which outputs a class label for a transient source that exists in the data. This approach, however, is not very useful for our application as it requires exactly one transient to be present somewhere inside each data segment and it does not allow for transients to span multiple segments. In our application, there is no guarantee that a transient signature is going to fit neatly into a time window regardless of it's size (due to unknown start and end times of the transient events) and it is almost guaranteed that there will not be a source present in every consecutive data segment.

Another recently proposed detection and classification method applies a set of hierarchical likelihood ratio tests to each data vector to classify it [15]. The parameters for these likelihood ratios are estimated using a Kalman Filter and the classification decisions for the individual data vectors are combined to classify the events that span over multiple observations. Although this sequential random coefficient tracking (SRCT) method was shown to have better performance when compared to a Gaussian Mixture Model (GMM)-based method [16–18], where the likelihoods of each observation vector under each hypothesis are modeled using a mixture of Gaussians, it still struggled with detecting sources in the presence of multiple interference signals as is sometimes the case in our particular application.

An extension of the previously mentioned CUSUM method is the Sparse Coefficient State Tracking (SCST) algorithm [19]. This algorithm is able to isolate sources from interference signals regardless of the number of interference signals present and it is also able to model much more complex sources than the SRCT method [19]. This method also makes use of probability ratio tests similar to the generalized likelihood ratio test (GLRT) [20] to detect presence of sources in the data as well as assign class labels to detected events. However, unlike SRCT and the HMM based methods, SCST uses a Bayesian Network (BN) model [21] for the sources of interest allowing for greater structural variation of the sources. This method also allows for many types of interference to be present simultaneously as it creates separate subspaces for source and interference classes and then nullifies the parts of the incoming signal that lie in the interference subspace, thus removing interference present in the signal, before detection and classification is performed. Although slightly more computationally demanding, this method was shown [22] to perform much better on acoustic data from national parks than the previously developed SRCT method and a standard GMM-based method.

1.3 Literature Review - Beamforming and Source Tracking

Once a source is detected and classified, the system will need to generate the trajectory of the moving source. Since only acoustic data is available for this purpose, a beamforming algorithm will need to be used to estimate the direction of arrival (DOA) angle of the acoustic source relative to the microphone array and these DOA estimates will then need to be used to create the trajectory of the moving source. Since all of the sources of interest occupy a large range of frequencies, a wideband beamforming algorithm will need to be employed for this purpose.

Wideband beamforming algorithms [23–26] use sensor array processing to determine the DOA of a signal based on the time delays at which each element in the array recorded the signal. Some of the popular wideband beamforming algorithms include wideband extensions of the multiple signal classification (MUSIC) algorithm [23], steered covariance matrix (STCM) algorithm [24], and weighted subspace fitting (WSF) algorithm [25].

MUSIC is an algorithm [27] used to find the DOA estimates of multiple sources from the spatial covariance matrix of the observations. It is thus inherently a narrowband beamforming technique though it can easily be extended to wideband DOA estimation using coherent or incoherent averaging methods [23]. Although this algorithm does perform better when estimating DOAs than simply picking peaks in power spectra of a beamformer, it requires the computation of eigenvalues and eigenvectors of spatial covariance matrices and also the a priori knowledge of the number of sources and structured interference present in the observed data. Since we do not have this a priori knowledge in our application and finding eigenvectors and eigenvalues of spatial covariance matrices would require far more computational resources than we can spare, the MUSIC algorithm is ill-suited for our application.

In STCM [24], the covariance matrices for each narrowband component are pre- and post-multiplied by transformation matrices before being summed together to form a single focused covariance matrix. This coherent beamforming method works better than incoherent averaging methods when the SNR is low. However, this method has additional computational requirements compared to incoherent averaging methods due to the need to compute and use transformation matrices when forming the focused covariance matrix [28]. Furthermore, it was shown [28] that under certain SNR conditions the incoherent methods produce more accurate DOA estimates for highly peaked spectra when compared to coherent methods so this additional complexity might not be warranted depending on the application.

The WSF method [25], on the other hand, finds the DOA estimates by solving a system of homogeneous non-linear equations based on the eignevalue decomposition of the narrowband covariance matrices. It involves a minimization problem which may contain multiple local minima and thus does not guarantee that the correct DOA angles will be found. This method requires more computations than the STCM method and also is not guaranteed to produce good results for the DOA estimates making it a poor candidate for implementation on our sensor board system.

To address the issues with the previously mentioned methods, the authors in [26] proposed an incoherent extension of the Capon beamformer to the wideband case using various averaging techniques including geometric, arithmetic, and harmonic averaging. They showed that the geometric averaging variant produced the narrowest beampattern out of all the wideband Capon variants thus allowing for the best DOA resolution of a source while maintaining the computational demands at a manageable level. This DOA estimation method is thus ideal for implementation on our sensor board as it requires the fewest computations out of all the previously mentioned methods while at the same time provides DOA estimates that are accurate enough for our purpose.

1.4 Contributions of Present Work

The main contributions of this thesis involve the implementation and verification of detection, classification and tracking algorithms on our custom-designed environmental monitoring system (EMS) that can be deployed in national parks. The EMS is an FPGA-based sensor platform that has built-in temperature, pressure, and light sensors for monitoring environmental parameters as well as an array of microphones which can be used for detecting, classifying, and tracking acoustic sources. This system also comes with support for a Global System for Mobile communication (GSM) modem and a low power wireless transceiver which can be added to the system via built-in expansion slots and used to route data from it to a central location for storage and analysis. This system is designed to be low-cost, low-power, and flexible so that a network of these EMS devices can be deployed in large numbers to cover a large area (or ideally the entirety) of a national park allowing NPS to carry out large scale monitoring of the noise profile and other environmental conditions in different national parks throughout the US.

The EMS can detect and classify sources based on audio data stream from one of the microphones (e.g., reference) as shown in Figure 1.1. Once a source is detected and identified, the EMS will run a wideband DOA estimation algorithm using acoustic data that was captured while the source was present. This algorithm uses data from all the microphones connected to the EMS in order to get successive DOA estimates for the classified source over the corresponding range of frequencies known for that source. These DOA estimates will then be logged to a file along with the start time, end time and class label of the source. This file will periodically be sent wirelessly to a base station for analysis using either an XBee radio or a GSM modem.

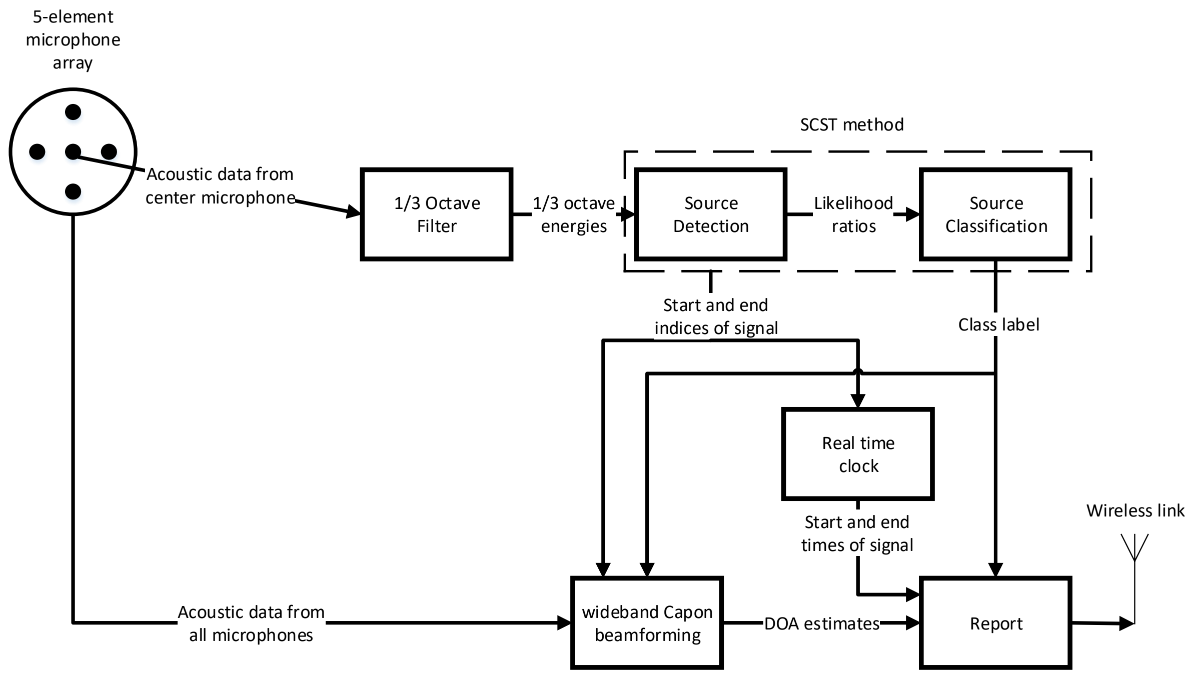

5-element microphone array
Acoustic data from center microphone
1/3 Octave Filter
1/3 octave energies
SCST method
Source Detection
Likelihood ratios
Source Classification
Start and end indices of signal
Class label
Real time clock
Start and end times of signal
Wireless link
Acoustic data from all microphones
wideband Capon beamforming
DOA estimates
Report

Figure 1.1: Overview of Acoustic Signal Processing on EMS Board

The detection and classification algorithm that was chosen for implementation on the EMS is the SCST algorithm [19] already developed and successfully tested for this problem. This particular algorithm was chosen because it was developed specifically for this type of application and was shown to provide superior results. This thesis outlines how this algorithm was partitioned and implemented on the EMS system and also provides validation results to show that this hardware implementation indeed provides comparable performance to the original MATLAB-based implementation of this algorithm.

The DOA estimation of the detected and classified sources will be performed using the geometric averaging wideband Capon beamforming algorithm [26]. This particular algorithm was chosen for its accuracy and ease of implementation on the FPGA-based system as discussed before. This thesis also reviews the theory behind this algorithm and outlines how it was implemented on the EMS system.

Since the EMS systems will be deployed in remote areas for an extended period of time they will be primarily battery powered. The EMS systems are designed to accept a range of voltages from 3.3 V all the way to 15 V so that a wide range of batteries can be used to power them. Furthermore, the boards can run off solar power and even have built-in charging circuitry so that the batteries can be recharged from a solar panel. This option allows for long-term deployment of these systems.

The EMS system running the SCST algorithm was tested at Indian Pass in Lake Mead National Recreation Area in California. It was first set up for 48 hours to simply capture 1/3 octave data. This data was then manually analyzed to extract sources of interest and interference which were then used to train the system. After that, the system was deployed in the same place again for 48 hours and performed on-board detection and classification of helicopters, jets and propeller planes in addition to recording 1/3 octave data.

Although the end result is to have detection, classification, and beamforming working in tandem, the beamforming code was tested separately in order to simplify the tests and better account for external variables such as the trajectories of the sources and number of sources present at one time. The beamforming test entailed tracking an electric model plane that was flown at a known trajectory relative to a microphone array attached to the EMS. The DOA angle estimates (namely azimuth angle of the plane relative to the array) made by the wideband Capon beamforming algorithm running on the EMS were then plotted over time and interpolated to create an estimated trajectory. This trajectory was then compared to the actual trajectory of the plane.

1.5 Thesis Organization

This thesis is organized as follows. Chapter 2 gives a brief overview of the legacy Larson Davis system currently used by NPS to capture acoustic data as well as the hardware on the EMS system and their function. Chapter 3 provides a brief summary of the SCST algorithm [19]. The implementation of this algorithm on the EMS system is presented in Chapter 4 while validation and test results of said implementation are detailed in Chapter 5. Chapter 6 gives an in-depth review of the wideband Capon beamforming method [26], the implementation of this beamforming algorithm as well as field test results. Finally, Chapter 7 summarizes the findings and observations presented in this thesis and outlines potential areas for future work related to the EMS system.

CHAPTER 2
HARDWARE SYSTEMS AND COMPONENTS

2.1 Introduction

Currently, NPS uses expensive Larson Davis 831 sound level meters (SLMs) to record 1/3 octave energy levels at various locations in the parks. These devices must be manually set up to record and then taken down to retrieve the data making them very labor intensive. Furthermore, the data must then be uploaded to a separate computer for manual analysis by expert operators who are trained to identify different types of sources from their time-frequency 1/3 octave signatures. Consequently, there is a large time gap between the time data is recorded and the time a soundscape for that park area is produced.

The EMS was designed to eliminate this long delay and streamline the process of soundscape characterization. Unlike the Larson Davis, the EMS is designed to record 1/3 octave energy data and process that data autonomously. The Artix 7 FPGA on the EMS printed circuit board (PCB) allows for near real-time implementation of detection and classification algorithms which can be used to automatically detect and classify sources of interest in the captured 1/3 octave data. Furthermore, the EMS system supports multiple microphone arrays allowing for a beamforming algorithm to be implemented in order to generate an approximate trajectory of the classified moving sources relative to the EMS. This extra position information could be invaluable in identifying flight routes across national parks as well as aggregating source data from different areas in a park. Also, the EMS is designed to eliminate the "man-in-the-loop" for soundscape characterization by reporting the classification and localization results to a central location in a park via low-power wireless transceivers and GSM modems which are added to the EMS system using built-in expansion slots. This automated reporting would eliminate the need for manual collection and setup of noise monitoring stations and thus would allow NPS to make quicker and more frequent soundscape updates. Finally, the low-cost, low-power and small form factor features of the

EMS system will allow for large scale deployment of these devices in national parks as well as other potential sites.

In this chapter, the capabilities of both the Larson Davis 831 sound level meter and the EMS system are discussed in depth. First, the current data acquisition setup used by NPS is outlined. Next, the capabilities and shortcomings of the Larson Davis 831 meter are outlined as pertaining to this application. Then, a brief overview of the EMS system hardware is given followed by detailed analysis of each of the subsystems on the EMS. The chapter concludes with a summary of the EMS system functionality and its intended operating protocol.

2.2 Larson Davis 831

Currently, in order to characterize soundscapes in national parks NPS sets up monitoring stations using commercial off-the-shelf (COTS) recorders to collect raw acoustic data in the form of MP3 files as well as 1/3 octave energies. The MP3 files are recorded by inexpensive audio recorders like the Roland R-05 while the 1/3 octave data is collected by the Larson Davis 831 sound level meter shown in Figure 2.1. Figure 2.2 shows a sample of the 1/3 octave data for helicopter source signature over a period of 101 seconds. The MP3 data is only used to help make classification decisions of an event found in the 1/3 octave data if it is difficult to make this decision just based on the 1/3 octave signature. Unlike the Roland recorder, the Larson Davis meter is very costly and does not have cheaper alternatives as 1/3 octave data recording is not a common application. At the same time, the Larson Davis meter offers reliable and high quality data recording with many additional features. It uses a detachable microphone and preamplifier allowing the user to choose a combination best suited for their application. The 1/3 octave digital filter bank used in the Larson Davis meter complies with all requirements of IEC 61260 Ed. 1.0 for Class 0 and all requirements of ANSI S1.11-2004 for Class 1 device which are the most stringent and second most stringent requirements for 1/3 octave filtering in their respective standards. This meter also supports a variety of hardware add-ons including a global positioning system (GPS) antenna, a GSM (Global

System for Mobile Communications) communication module, and a sensor suite capable of recording wind speed, wind direction, temperature, and humidity. The problem with these add-ons is that they are expensive and require their own power sources (e.g., powered USB hub). This makes their use impractical for environmental monitoring scenarios in remote locations where everything must run on batteries.

Figure 2.1: Larson Davis 831 Sound Level Meter

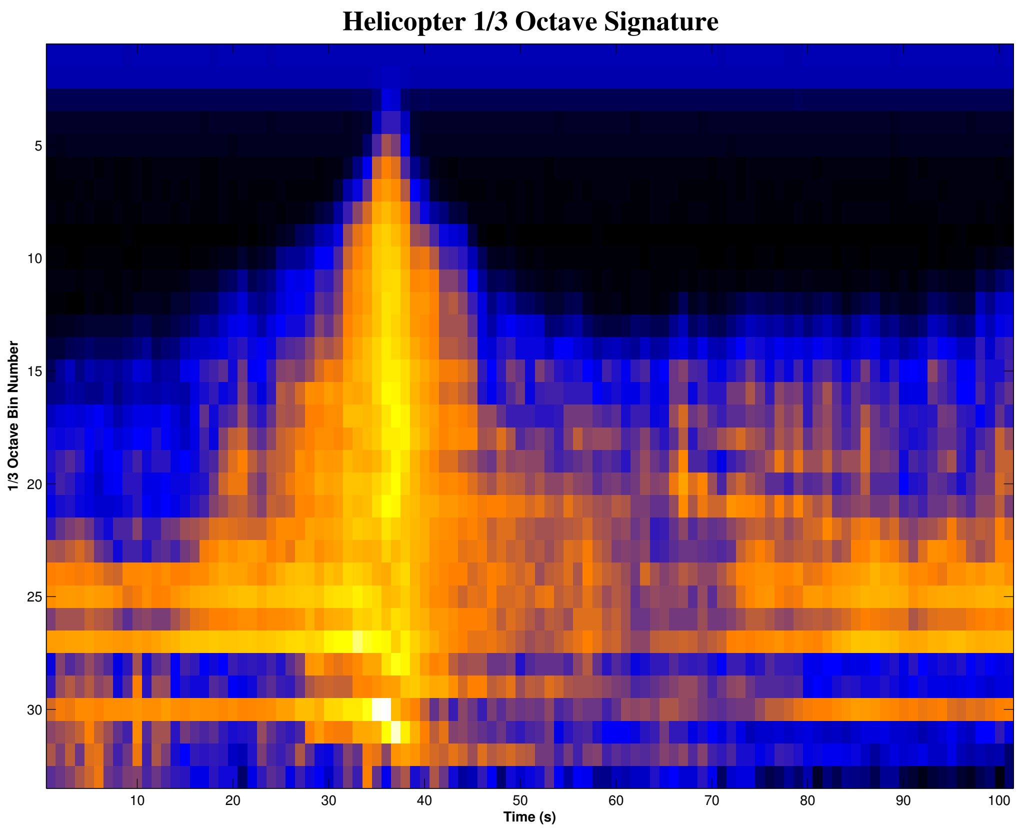


Figure 2.2: Helicopter Signature in 1/3 Octave Data

The Larson Davis meter has several additional drawbacks when used for monitoring remote locations in national parks. First of all, it must be manually set up at each site before it is able to record 1/3 octave data to on-board memory. Furthermore, since the 1/3 octave data is stored locally and the Larson Davis meter has no power efficient wireless communication capabilities, the data can only be retrieved manually after stopping the recording process and bringing the meter to a lab for analysis. This is obviously not ideal for continuous monitoring as the system is not operational while the data is being downloaded. Furthermore, since there is no way to check the data integrity and fidelity until someone physically comes by to pick up the meter, any problems that might have occurred during the recording phase (e.g., the system got knocked over by wind or the battery died) will not be detected until the end of the recording. This creates the potential for a lot of the data to be compromised during a recording run.

Because the Larson Davis only records 1/3 octave data, all the analysis that needs to be done on the data in order to locate and classify sources is currently done in post-processing. As previously mentioned, this post-processing is manual and is a very labor intensive process. It takes much longer to parse through the data manually than it does to collect it creating a huge bottleneck in the soundscape characterization process which not only creates a long delay between data acquisition and a soundscape profile being generated but also means a lot of the collected data never gets used. This in turn prevents NPS from having an up-to-date soundscape profile for many areas that they monitor in national parks. The delay between gathering data and detecting and characterizing sources in that data reduces the effectiveness of any actions the NPS would like to take based on this data such as managing air tours and traffic over the parks. Even if NPS had the manpower to sift through all the data manually, the fact that the monitoring systems need to be periodically taken down in order to access the recorded data would create periods of time when data is not available for some monitoring locations. This makes it likely that some sources would not be captured and thus the accuracy of the soundscape profiles computed for these locations would be compromised.

2.3 EMS Hardware Overview

As mentioned previously, the EMS system was custom-designed for this particular application and as such its hardware was specifically selected to address the issues mentioned in the previous section. Aside from the sensor suite and microphone ports, the EMS system also contains an Artix 7 FPGA which implements all the processing and glue logic for the data collected by the sensors as shown in Figure 2.3. In addition, the EMS has several components for storing data, time synchronization, GPS localization, charge controller circuitry, a battery gas gauge, a microcontroller, a USB debug port, 2 expansion connectors for wireless communications and a JTAG port for programming the FPGA. All these components (shown in Figures 2.4 and 2.5) work in tandem to provide timely, in-depth reports of the environments in which the EMS systems are deployed to the NPS technicians and/or park rangers. These reports can be used to easily create soundscapes of these areas without having to manually analyze acoustic data. These key components are reviewed in the following subsections.

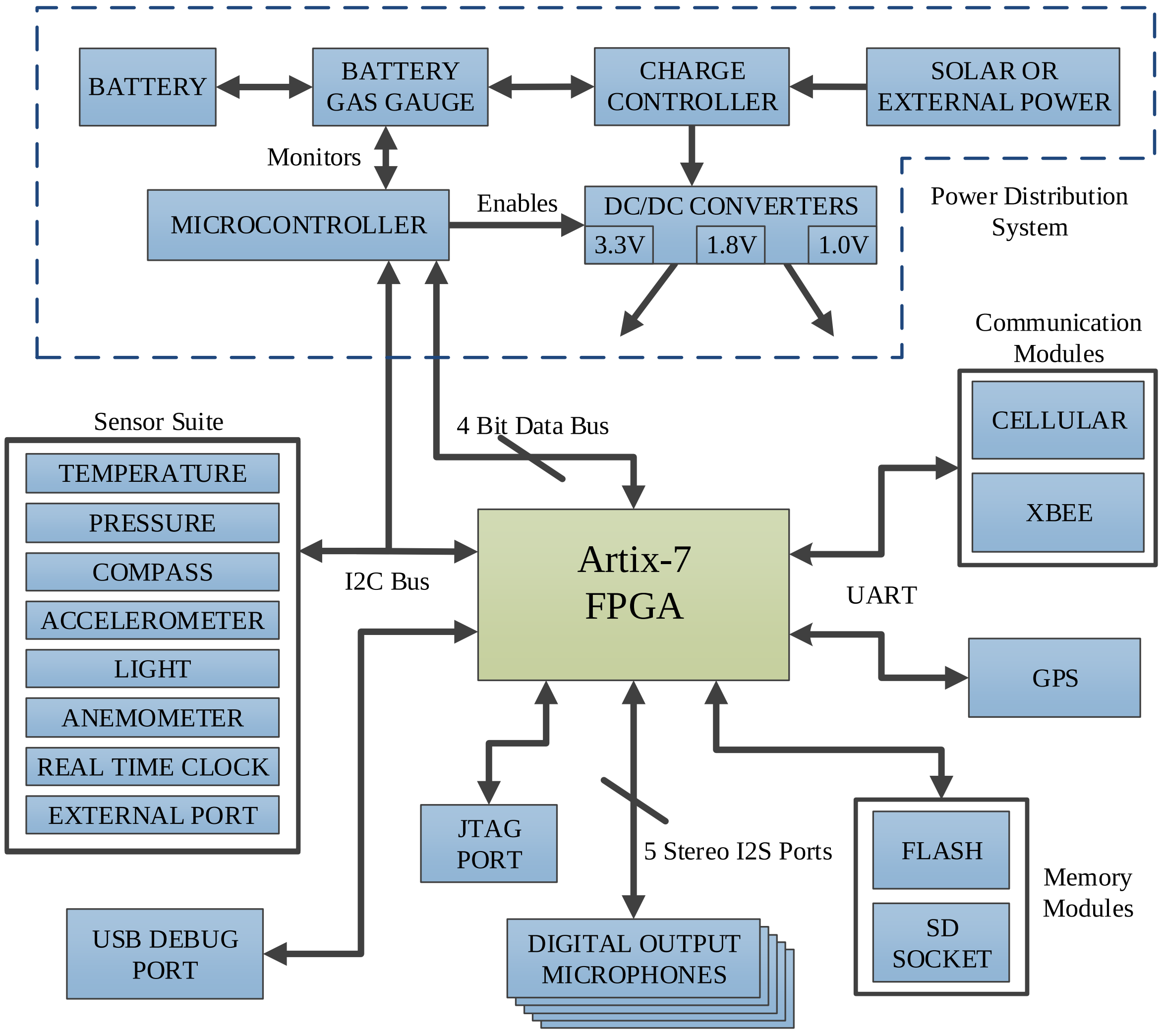


Figure 2.3: EMS Components and Connections

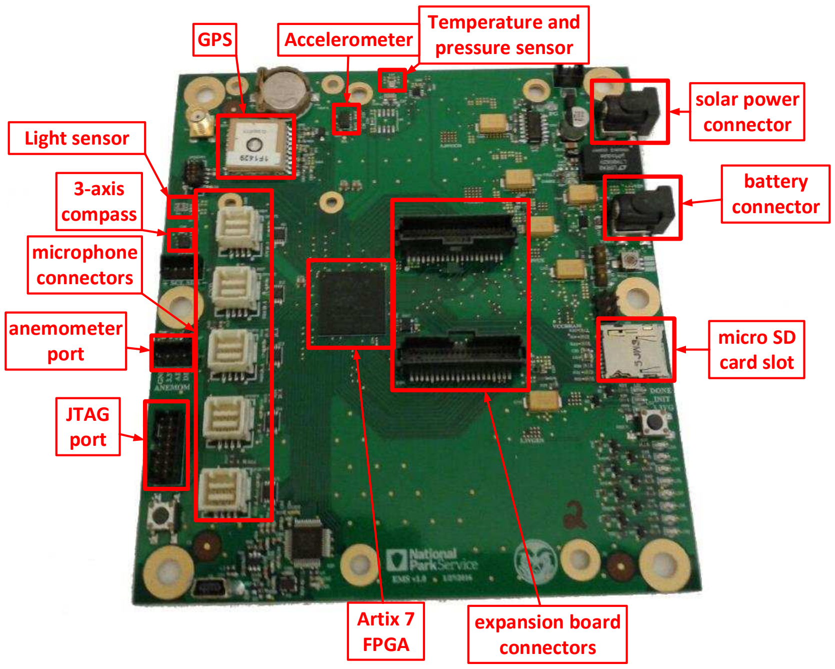


Figure 2.4: EMS PCB Top View

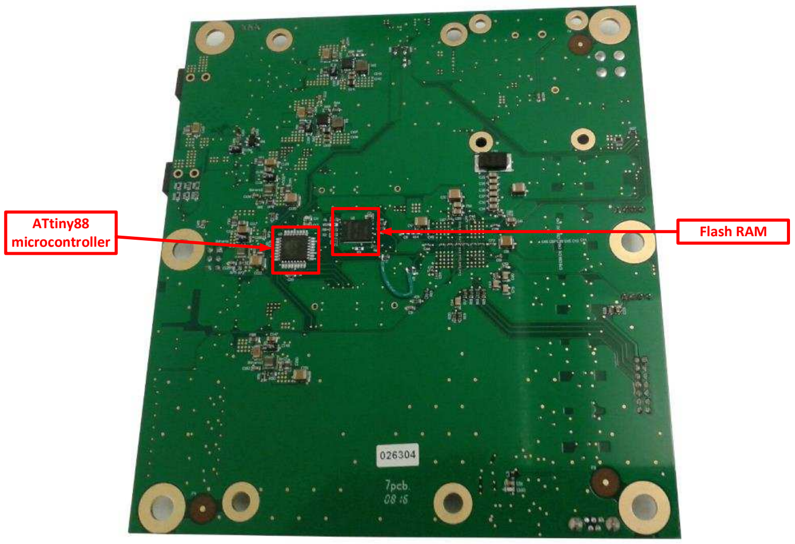


Figure 2.5: EMS PCB Bottom View

2.3.1 Artix 7 processing core

The main component of the system which routes the signals between the other components and does all the data processing is the Xilinx Artix 7 FPGA. The FPGA implements all the drivers for the various sensors and microphones attached to the board allowing for easier modification of the design should some components be replaced. The Artix 7 also performs all of the 1/3 octave filtering of the acoustic data and subsequent detection, classification and tracking of important sources. This allows for easy modification of the filters and processing algorithms for future upgrades. For instance, if a different data format for detection were desired, the 1/3 octave filter bank can easily be replaced with a filter bank implementing, for example, a wavelet transform or some other logic without changing the design of the board. The Artix 7 was selected because at the time of writing this document, it is the cheapest and lowest power FPGA that is still large enough to accommodate the glue logic and filtering necessary for the EMS system. The xc7a100tftg256-2 part which was used in the actual design has 608 kB of RAM and 101,440 logic cells. These resources are sufficient for the current design and also leaves room for implementation of other features on the EMS system as the current design only uses about half of the available resources. This part also has 300 digital IO ports allowing for interfacing to the flash memory, SD card and all the other peripherals on the board directly without a need for shared buses. This FPGA can also run at up to 100 MHz which is more than fast enough for processing acoustic data.

2.3.2 Non Volatile Memory

Since the Artix 7 FPGA used in the EMS system only has 608 kB of RAM, an additional 4 MB FLASH memory chip was added to the EMS. This Flash memory is used to store both the bitstream used to configure the FPGA at power up and also the various parameters (e.g., log-likelihood values) which are used by the SCST algorithm.

In addition to the Flash memory, the EMS system also has a micro SD socket which allows the Artix 7 FPGA to communicate with a standard SD card using the SPI protocol. This card can be used as backup storage for acoustic event reports in case wireless connectivity is not available to send these reports directly to a base station. The SD card can also store system configuration parameters which can be used to change system operations based on the application.

2.3.3 Environmental Sensors

In addition to acoustic monitoring, the EMS can also collect general purpose data about the environment in which it is deployed via a range of on-board sensors. It has a BMP280 barometric pressure and temperature sensor, a separate port for an LM75B temperature sensor that can be placed on the outside of the system enclosure, a HMC5883L 3-Axis Compass, an ISL 29023 light sensor, an ADXL345 accelerometer and a port for a Davis Instruments 07911 anemometer. The compass will be able to provide a precise orientation of the system and microphone array to the beamforming algorithm in order to produce accurate DOA estimates. The accelerometer will be able to detect the orientation of the system and identify if it has been knocked over by wind or an animal. The light sensor can be used for power management by powering down some non-essential systems when there is not much sunlight available to operate the system and charge the battery. The temperature, pressure and anemometer will provide useful information about the environment of the park which will be reported along with the acoustic detection and classification results to a central base station. Additionally, these in-situ measurements can be used to estimate speed of sound depending on temperature and wind velocity profiles.

2.3.4 Communication Systems

The EMS system is designed with 2 expansion slots for wireless communications: one for a cellular module (U-Blox SARAU260) and the other for any XBee radio transceiver. This allows for flexibility in deployment of the systems as either the GSM module or the XBee radio can be used to transmit data from the EMS to a central base station depending on the availability of cellular service in the deployment location and the distance between the system deployment site and the park base station. If multiple EMS systems are deployed, the XBee radios can be used to route data from systems deployed in remote locations with poor cellular connectivity to one or a few systems that have good cellular connectivity. These systems will then send their own reports plus those routed to them via XBee radios to a base station. This routing-scheme would reduce the power consumption and cost of the EMS network as only a few boards which will communicate directly with the base station will potentially need to have a cellular modem installed. In this routing scheme, the number and location of the boards with cellular modules will be dictated by the size of the overall network, bandwidth of the cellular modules and the strength of the cellular signal at the different locations in the park that will be monitored.

2.3.5 Power System

The EMS is intended to be deployed in remote locations and is therefore designed to be powered by batteries. It has a barrel jack input for DC power and can accept a range of voltages from 3.3V to 5V. This provides the user flexibility in selecting batteries to power the system for the duration of its deployment.

For particularly long deployments, the EMS system is also able to accommodate a renewable energy source like solar power. It has a separate barrel jack connection which can be used to connect to a photovoltaic solar panel that provides sufficient voltage and current to power the system. Furthermore, the EMS system has charge controller circuitry so that the excess power from the solar panel can be used to recharge the battery. With the solar panel setup, if there is enough insolation during the day, the EMS system can operate continuously for a long time by using solar power during the day and relying on battery power at night. Using on-board switches, battery float voltage can be adjusted, maximum peak power tracking can be enabled for solar panels, and charge termination can be adjusted. These are necessary to allow for a broad range of battery chemistries and input power combinations. To achieve minimum current draw and proper input/output states of the FPGA during power on, the system DC/DC converters are enabled in a particular sequence according to the switching characteristics specified for this FPGA by the manufacturer. An Atmel ATtiny88 microcontroller was selected to implement the power supply sequencing. This controller can also monitor the amount of battery charge remaining via an on-board battery gas gauge. This information can be used to put the system in sleep mode if the battery charge remaining gets critically low during minimal insolation periods. The microcontroller and FPGA are linked via a 4 bit communication bus so that data can be easily shared between the two.

2.3.6 Time Synchronization and Localization

The EMS system also has a GPS module for time synchronization and node localization. The time from the GPS module is read after acquiring a fix and is used to set the real-time clock on the EMS system. This real-time clock is then periodically synchronized to the time reported by the GPS in order to prevent clock drift and should limit synchronization error to the precision of the GPS module (10 ns for the module used in the EMS system). This real-time clock in turn is used to timestamp acoustic events that are detected by the Artix 7 and reported to the base station. Precise time synchronization is necessary not only for DOA estimation but also for merging data between EMS systems deployed in a park. If a GPS signal is not available in some deployment locations but time synchronization is still necessary, then a network based synchronization protocol such as the Flooding Time Synchronization Protocol (FTSP) [33] will need to be implemented on the EMS systems.

The GPS will also be able to provide accurate location information (to within 3 meters of the receiver) for each of the deployed nodes. This information will be integrated into the reports sent by each EMS system allowing NPS technicians to easily identify where each acoustic event occurred and to visualize the network of deployed boards on a map. The accuracy of location finding will affect the accuracy of trajectory generation of the classified sources.

2.4 Microphone Array

The EMS system has been designed with 5 digital audio channels each of which is able to support up to 2 digital microphones in a stereo configuration. This configuration allows the system to capture acoustic data from up to 10 different microphones simultaneously, if needed. This enables the system to capture data from an array of microphones (of any geometry) for beamforming and direction of arrival (DOA) estimation of moving sources. For the tests presented in this thesis a simple 5 microphone wagon-wheel array with a radius of 98 mm, where 4 microphones are spaced evenly around the perimeter of the array and 1 microphone is in the center, was manufactured and populated with InvenSense ICS-43432 digital output microphones. These microphones were selected because of their large dynamic range and low cost. Figure 2.6 shows the top view of the microphone array in which the mounting hole and acoustic ports (which are connected to the microphones) can be seen while Figure 2.7 shows the bottom view in which the microphone adapter boards and mounting flange can be seen. A custom adapter PCB was developed for the microphones which included connectors, buffers, and terminations for the signal lines. These microphone adapters were used in the construction of the array as seen in Figure 2.7. The array shape was decided based of it's ease of manufacturing and characterization. Since the drivers for the microphones are implemented on the Artix 7 FPGA, most types of digital output microphones can be used with only minor modifications to the HDL code. This allows flexibility in selecting microphones with characteristics suited for specific applications. The wagon-wheel array allows for accurate azimuth angle DOA estimation although it is not able to accurately estimate the elevation angle of incident acoustic signals due to it's planar shape. This could be improved by elevating some of the microphones (e.g., the one in the center).

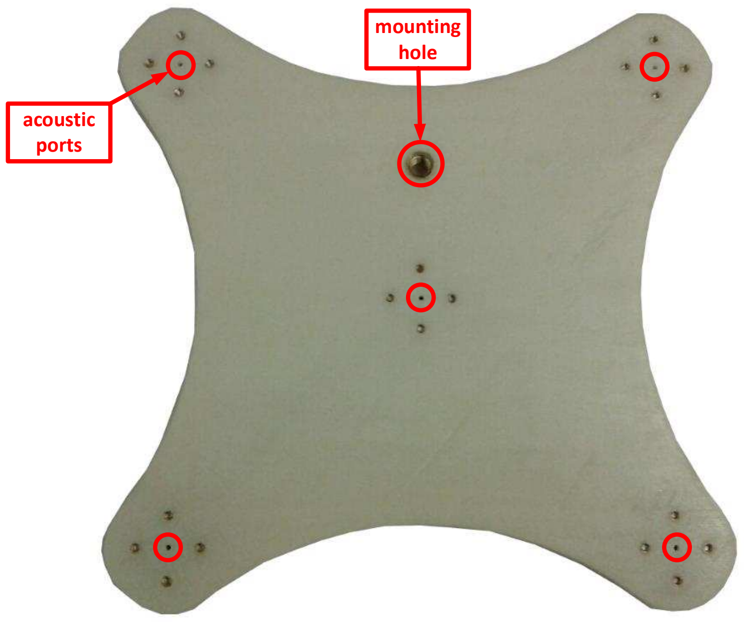


Figure 2.6: Microhpone Array Used In Tests Top View

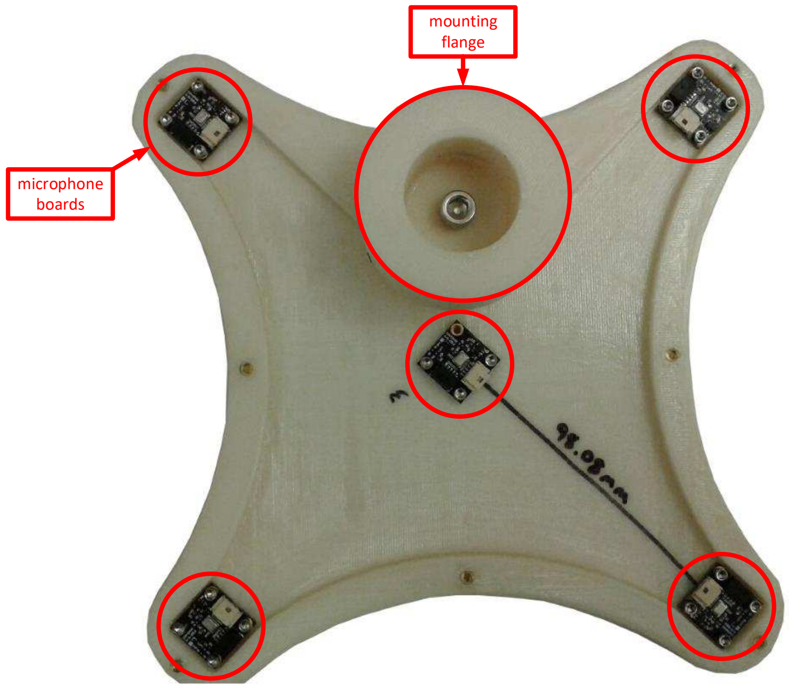


Figure 2.7: Microhpone Array Used In Tests Bottom View

2.5 EMS Operating Protocol

When deployed in a park the typical operation of the system will be as follows. First, after it is set up in a location by an NPS technician, the system will boot up, program the FPGA and calibrate the real-time clock based on GPS data. Then, it will start recording 1/3 octave data to the SD card while simultaneously looking for sources of interest (e.g., propeller planes, jet) in the streaming 1/3 octave data. Once it identifies the presence of one of these sources it will compute spatial covariance matrices for all frequencies which are used by at least one source type every second while the source is present and temporarily store these matrices on the SD card. Then, once the source is classified, the system will load those buffered matrices and use the ones that are relevant to the identified source type to perform DOA estimation for every second during which that source was present. After this is done, the system will record the start and end time of the source signal (based on the real-time clock), the class of the source and the sequence of DOA estimates that were made for this source to a log file on the SD card. In addition, the system will periodically record environmental data (e.g., temperature and pressure) from the onboard sensors to this log file. This log file will be sent to a central base station for further analysis via a cellular modem or an XBee radio depending on the reception of the system. After the file is sent, a new file will be created for the next time period and old log files will be erased as needed from the SD card in order to prevent it from filling up completely.

2.6 Conclusion

The Larson Davis system, while able to provide high accuracy 1/3 octave data, is not suited for long-term environmental monitoring. It is expensive, with no built-in wireless communication options and uses a lot of power. On the other hand, the EMS system is streamlined to collect acoustic data, perform near real-time processing on the data and then send out periodic reports about the soundscape environment to a central base station for further analysis by NPS specialists. To this end, it has a powerful FPGA chip which can process acoustic data collected by the microphones with minimal delay and save the processed data along with environmental data collected by the onboard sensors to log-files on an SD card. Then, it can periodically send out these log-files to a central base station via a cellular modem and/or an XBee radio which can be easily interfaced with the EMS system. The EMS systems can therefore be deployed in multiple locations in a park for extended periods of time for autonomous environmental monitoring. These systems will allow NPS to have accurate, up-to date statistics about the soundscapes and environmental conditions in many natural areas. The only downside of the EMS when compared to the Larson Davis 831 meter is that it does not provide 1/3 octave data as accurate as the Larson Davis because it uses much cheaper microphones and has no pre-amplifier or analog filters. However, since the 1/3 octave data is primarily used for detection and classification of acoustic sources and the EMS can do that even on the less accurate 1/3 octave data, this loss in accuracy is not a significant concern. This is particularly true since the system is trained on the data that it captures itself so, unless the loss in 1/3 octave data accuracy results in a loss of discriminatory information present in the observations, the detection and classification performance of the EMS system should not be negatively impacted.

CHAPTER 3
REVIEW OF SCST

3.1 Introduction

The Sparse Coefficient State Tracking (SCST) algorithm [19] performs detection and classification of transient acoustic events. The algorithm does this by continuously monitoring the incoming acoustic data and detecting the start and end of a source. Once the end of a source is detected the algorithm is able to consider the source over its entire duration to identify the type of the source that was present. This type of signal detection and classification avoids many of the drawbacks of the popular windowing methods [9–11, 14] where time series data is broken up into sections and detection and classification is done on the windows independently. With this method, there is no need to fiddle with window length and there is no need to do any kind of decision fusion at the end (e.g., to decide whether 2 consecutive windows detected 2 different signals or the same signal).

The entire detection and classification process of SCST can be broken down into several steps as shown in Figure 3.1. First, the raw observation vector is sparsely coded using a dictionary matrix which is composed of source and interference dictionaries and the elements which correspond to columns from interference dictionary matrices are discarded. This step attempts to separate out interference from the sources present in the observation by removing the elements in the vector which contain interference information only. The resulting vector is then quantized according to pre-determined quantization levels and the probabilities of this quantized vector given the quantized vector from the previous sample are computed under each source and noise model using Bayesian Networks. The log likelihood ratios for each source are then computed and added to their respective running sums. These running sums are compared against pre-selected thresholds to determine start and end times of the transient sources. Once the end of a source signal is detected the classification is done based on the same cumulative sums.

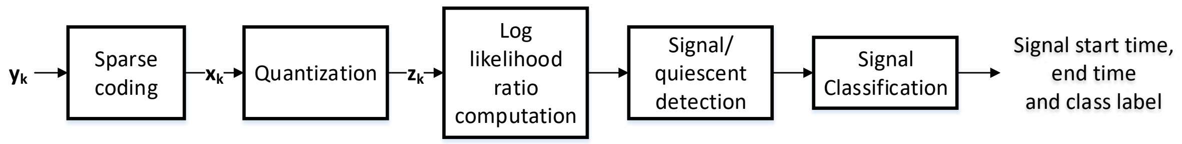

y_k
Sparse coding
x_k
Quantization
z_k
Log likelihood ratio computation
Signal/ quiescent detection
Signal Classification
Signal start time, end time and class label

Figure 3.1: SCST Data Flow Diagram

In this chapter, the SCST algorithm is reviewed starting from the pre-processing steps of feature extraction and quantization and ending with a discussion on the optimal implementation of the algorithm. The chapter concludes with a summary of the way SCST performs detection and classification as well as some of the benefits/drawbacks of this method.

3.2 SCST Framework

3.2.1 Feature Extraction and Quantization Processes

As the data from the central microphone in the array is obtained it is passed through a 1/3 octave filter bank to produce 33 frequency band signals. These signals are aggregated every 1 second and their energies are computed. These 1/3 octave energy vectors form the data matrix $\mathbf{Y} = [\mathbf{y}_1 \ldots \mathbf{y}_n]$ where each column $\mathbf{y}_k$ contains the energies from each of the 33 bands at time k.

The SCST algorithm then sparsely codes each observation vector $\mathbf{y}_k$ using a set of disjoint dictionaries obtained using KSVD [35] with one dictionary $\mathbf{H^p}$ per source type, p, and one more dictionary matrix for all types of interference cases. KSVD constructs signal-dependent dictionaries with a user-specified number of columns. These dictionaries are computed to optimally (in the mean squared error sense) reconstruct the training data using the fewest number of dictionary atoms to reconstruct each training vector. In other words, KSVD solves the following optimization problem

$$\min_{\mathbf{H}^p, \mathbf{X}^p} ||\mathbf{Y}^p - \mathbf{H}^p \mathbf{X}^p||_F^2 \quad \text{subject to} \quad \forall i, ||\mathbf{x}_i||_0 \leq \tau \tag{3.1}$$

where $\mathbf{H}^p$ is the dictionary specific to the p^{th} source class, $\mathbf{Y}^p$ are all the training samples for that class, $\mathbf{X}^p$ are the sparse codes that represent those samples, and τ is a user defined sparsity parameter.

KSVD does this by alternating between two optimization phases. First, it starts with a random dictionary for a single signal class $\mathbf{H}^p$ and solves (3.1) for the optimal $\mathbf{X}^p$, using a pursuit algorithm like Basis Pursuit Denoising (PBDN) [36] or Orthogonal Matching Pursuit (OMP) [37], while keeping $\mathbf{H}^p$ fixed. Then, it uses these newly found codes to update the dictionary $\mathbf{H}^p$ one column at a time. It first forms a restricted error matrix $\mathbf{E}^p_k$ for a dictionary column $\mathbf{h}^p_k$ which represents the reconstruction error when not using column $\mathbf{h}^p_k$ and then uses Singular Value Decomposition (SVD) [38] on that matrix to find an update for $\mathbf{h}^p_k$ that would minimize the Mean Squared Error (MSE) when reconstructing $\mathbf{Y}^p$ from $\mathbf{X}^p$ and $\mathbf{H}^p$. This is done for each of the columns in the dictionary and then the algorithm switches back to the sparse coding phase. It alternates this way between the two phases until a local minimum for (3.1) is reached. Varying the number of dictionary atoms and the sparsity parameter allows KSVD to represent many different kinds of signals. For our tests we found experimentally that using 30 atoms for each dictionary and a reconstruction error threshold of 10^{-4} produced the best detection and classification results.

The sparse coding, during KSVD and when sparsely coding observations for SCST, is done using a modified version of the orthogonal matching pursuit (OMP) algorithm called Fast OMP [39] which does not require any matrix inversion operations. Like the original OMP, this is a recursive algorithm that solves the following equation:

$$\mathbf{x}_k = \arg\min_{\mathbf{x}} ||\mathbf{y}_k - \mathbf{H}\mathbf{x}||_2 \quad s.t. \quad ||\mathbf{x}||_0 \leq \tau \tag{3.2}$$

by finding the optimal sparse vector $\mathbf{x}_k$ which produces the smallest (in MMSE sense) reconstruction error in representing $\mathbf{y}_k$.

Just like the original OMP algorithm, the fast OMP in [39] starts by finding the dictionary column from dictionary $\mathbf{H} = [\mathbf{h}_1 \ldots \mathbf{h}_E]$ that has the highest inner product with the observation vector $\mathbf{y}_k$

$$\mathbf{h}_{e_1} = \arg\max_e |\mathbf{y}_k^T \mathbf{h}_e| \tag{3.3}$$

This column vector forms the first column of matrix $\tilde{\mathbf{H}}_1$ (which at this step only has this one column). Then this matrix is used to create a least squares filter according to the following equation:

$$\mathbf{Q}_{\tilde{\mathbf{H}}_1} = \mathbf{q}_1 = \frac{\mathbf{h}_{e_1}}{||\mathbf{h}_{e_1}||^2} \tag{3.4}$$

This least squares filter is then used to filter the observation $\alpha_1 = \mathbf{q}_1{}^T \mathbf{y}_k$ and the resulting value is inserted into the first entry of the result vector $\hat{\mathbf{x}}_1 = \alpha_1$. The residual vector is computed using

$$\mathbf{r}_1 = \mathbf{y}_k - \alpha_1 \mathbf{h}_{e_1} \tag{3.5}$$

After this initialization, the algorithm repeats in much the same way but this time uses the residual vector $\mathbf{r}_{t-1}$ from the previous iteration $(t-1)$ instead of the observation vector. A new iteration starts by finding the dictionary column $\mathbf{h}_{e_t}$ that has the maximum inner product with the residual and which has not been selected in a previous iteration

$$\mathbf{h}_{e_t} = \arg\max_e |\mathbf{r}_{t-1}^T \mathbf{h}_e| \tag{3.6}$$

Then, this column is filtered with the least squares filter computed at the previous iteration as follows:

$$\mathbf{b}_{t-1} = \mathbf{Q}_{\tilde{\mathbf{H}}_{t-1}} \mathbf{h}_{e_t} \tag{3.7}$$

After that, the portion of the newly selected column which does not lie in the subspace spanned by the previously selected columns is computed using:

$$\tilde{\mathbf{h}}_{e_t} = \mathbf{h}_{e_t} - \tilde{\mathbf{H}}_{t-1} \mathbf{b}_{t-1} \tag{3.8}$$

Then, a least squares filter is formed using this part of the newly selected column according to:

$$\mathbf{q}_t = \frac{\tilde{\mathbf{h}}_{e_t}}{||\tilde{\mathbf{h}}_{e_t}||^2} \tag{3.9}$$

This filter is then used to compute a value in the sparse vector for the currently selected dictionary column using:

$$\alpha_t = \mathbf{q}_t{}^T \mathbf{y}_k \tag{3.10}$$

The entire result vector is then updated to take into consideration the effect of this new dictionary column as follows:

$$\hat{\mathbf{x}}_t = \left[\hat{\mathbf{x}}_{t-1}^T - \alpha_t \mathbf{b}_{t-1}{}^T,\ \alpha_t\right]^T \tag{3.11}$$

Finally, the least squares filter is updated according to the following equation:

$$\mathbf{Q}_{\tilde{\mathbf{H}}_t} = \left[\mathbf{Q}_{\tilde{\mathbf{H}}_{t-1}}^T - \mathbf{q}_t \mathbf{b}_{t-1}{}^T,\ \mathbf{q}_t\right]^T \tag{3.12}$$

Then the currently selected dictionary column is added to the set of selected columns $\tilde{\mathbf{H}}_t = [\tilde{\mathbf{H}}_{t-1}\mathbf{h}_{e_t}]$ and the residual is updated by accounting for the contribution of the newly selected column to the reconstruction of the original observation as follows:

$$\mathbf{r}_t = \mathbf{r}_{t-1} - \alpha_t \tilde{\mathbf{h}}_{e_t} \tag{3.13}$$

This process is repeated as necessary until either the norm of the residual falls below some user defined threshold or a user defined number of columns has been selected for reconstruction. After this algorithm terminates, the $\hat{\mathbf{x}}_t$ vector is changed into the sparse vector $\mathbf{x}_k$ by rearranging the elements so that they align with their respective columns in the full dictionary $\mathbf{H}$ while the rest of the elements in $\mathbf{x}_k$ are filled with zeros. In our tests, we used the terminating condition of the norm squared of the error falling below 10^{-4} which was experimentally found to produce good sparse vectors for detection and classification.

Once the sparse coding is done, the elements of the sparse vector for each $\mathbf{y}_k$ corresponding to the interference atoms are mostly separated from those of sources and hence can be removed. This entire sparse coding step is done in the hopes that if any interference events were present in the observation vector they would be encoded using these particular elements in the sparse vector and so by removing them any influence interference would have on the sparsely coded observations would be eliminated. The resulting sparsely coded vectors $\mathbf{x}_k$ form the feature matrix $\mathbf{X} = [\mathbf{x}_1 ... \mathbf{x}_n]$ which will then be applied to the quantizer (see Figure 3.1.

Each $\mathbf{x}_k$ in matrix $\mathbf{X}$, is then converted into a *state vector* using an optimal quantizer designed specifically to increase class separability [40]. This quantization process allows for modeling the state evolution of a given class of source events using an indexed system for navigating a state transition model using Bayesian networks [21]. Additionally, it makes the implementation of the transient detector-classifier much easier and more effective while reducing the effects of ambient noise. Determining the quantizer's optimum transition levels involves solving a quadratic programming problem which can be done off-line during the training phase. The *coefficient state vector* is generated by quantizing the sparsely coded features in $\mathbf{x}_k$ for every column of matrix $\mathbf{X}$. The quantization process allows the features $x_k(i)$s (i.e. elements of $\mathbf{x}_k$) to be parameterized using a family of categorical distributions. The number of quantization levels L is chosen such that the quantized states evolve adequately for inter-class discrimination yet L is also kept relatively small since the quantizer resolution drives the state distribution model size. Our experimental results indicated that a quantizer with $L = 4$ levels is a suitable choice for this problem to make sure that the quiescent period following an acoustic source is accurately detected.

The $L-$level quantization function executed at runtime is simply

$$z_k(i) = \begin{cases} 0, & |x_k(i)| \leq \epsilon \\ Q\left(x_k(i)\right), & \text{otherwise} \end{cases}, \quad i \in [1, J] \tag{3.14}$$

where $Q(x) = l$ if $x \in (t_{l-1}, t_l]$, $l \in [1, L-1]$ and threshold ϵ is selected (during the training) to force coefficients likely attributed to ambient background noise into a single zero state. The transition levels t_l, $l \in [1, L-1]$ are also determined during training in order to preserve as much discriminative information as possible after quantization. The choice of using a subband-based quantizer verses a global quantizer (i.e. one for all $i \in [1, J]$) appeared to be a trade-off between inter-class discrimination and background noise rejection.

3.2.2 Detection and Classification of Transient Events

Separating the transient events of interest from the background noise and otherwise non-interesting data requires two distinct phases of operation: 1) signal detection to locate the presence of a transient signal of an unknown source under the assumption that none were present and 2) quiescent detection to find the endpoint of the transient signal by searching for observations where the particularly dominant source is no longer present under the assumption that there was one present. The approach assumes that each unique transient has a finite extent and is separated by a distinct, albeit variable length, quiescent period. If the quiescent period is not detectable, the source signals will be merged into a single event and could lead to classifier confusion.

When the data has been in a quiescent period since time $\hat{k}_0$, signal detection and classification can be accomplished on each new observed $\mathbf{z}_k$ using the following multiple hypothesis test to determine if it captures any potential source signal of interest,

$$\mathcal{H}_0 : \mathbf{z}_k = \mathbf{w}_k, \quad \hat{k}_0 \leq k \leq n \tag{3.15}$$

$$\mathcal{H}_1^{(p)} : \mathbf{z}_k = \begin{cases} \mathbf{w}_k, & \hat{k}_0 \leq k \leq k_1 \\ \mathbf{s}_k^{(p)} + \mathbf{w}_k, & k_1 \leq k \leq n \end{cases}$$

where $\mathbf{s}^{(p)}$ is a source vector of an unknown class $p \in [1, P]$ with P being the total number of considered source classes, and $\mathbf{w}_k$ is an ambient background noise vector. The null hypothesis $\mathcal{H}_0$ and alternative hypothesis $\mathcal{H}_1^{(p)}$ represent the absence and presence of any prominent transient source labeled p, respectively. The times k_0 and k_1 denote the onset times for the next unknown quiescent period and source event, respectively while $\hat{k}_0$ and $\hat{k}_1$ denote the estimates (generated by SCST) of the most recently observed quiescent and detection periods, respectively.

Given that the acoustic transient event has a finite extent, quiescent detection is needed to identify the end point of the transient source or onset of the next quiescent period. When a source signal has been present since time $\hat{k}_1$, the following hypothesis test can be used to perform quiescent detection,

$$\mathcal{H}_1^{(p)} : \mathbf{z}_k = \mathbf{s}_k^{(p)} + \mathbf{w}_k, \quad \hat{k}_1 \leq k \leq n \tag{3.16}$$

$$\mathcal{H}_0 : \mathbf{z}_k = \begin{cases} \mathbf{s}_k^{(p)} + \mathbf{w}_k, & \hat{k}_1 \leq k \leq k_0 \\ \mathbf{w}_k, & k_0 \leq k \leq n \end{cases}$$

i.e., $\mathbf{s}_k^{(p)}$s cease to be extant at the unknown time k_0 under the null hypothesis $\mathcal{H}_0$.

To implement the hypothesis test in (3.15) on streaming real-time quantized data vectors, $\mathbf{z}_n$, and provide a test statistic for signal detection and evaluating the relative likelihoods of $\mathcal{H}_1^{(p)}$, we adopt a cumulative test statistic formulation similar to the CUSUM procedure [13]. For the SCST method, this online test statistic is given by

$$B_p(n) = \max\{0,\ B_p(n-1) + b_p(n)\}, \quad n = \hat{k}_0, \hat{k}_0 + 1, \ldots \tag{3.17}$$

and initialized as $B_p(\hat{k}_0 - 1) = 0,\ \forall p$. This statistic is updated at time n using

$$b_p(n) = \begin{cases} \ln\left(\dfrac{f_{\lambda_p}(\mathbf{z}_n|\mathbf{z}_{n-1})}{f_{\lambda_0}(\mathbf{z}_n)}\right), & B_p(n-1) > 0 \\ \ln\left(\dfrac{f_{\lambda_p}(\mathbf{z}_n)}{f_{\lambda_0}(\mathbf{z}_n)}\right), & B_p(n-1) = 0. \end{cases} \tag{3.18}$$

where $f_\lambda(.)$ is a probability distribution modeled by $\lambda \in \{\lambda_0, \lambda_p\}$ containing noise, λ_0 and source,λ_p, parameter sets. When λ_p is known, this test represents the *sufficient statistic* for a binary hypothesis test between $\mathcal{H}_0$ and $\mathcal{H}_1^{(p)}$. However, since λ_p is unknown, our test determines which unknown source parameter set is the most likely one,

$$\max_p B_p(n) \geq \eta \tag{3.19}$$

i.e., the cumulative value of $B_p(n)$ simply must exceed the detection threshold η for any source label p.

Similarly, the quiescent detection phase uses the following cumulative test statistic for streaming data,

$$T_p(n) = \max\{0,\ T_p(n-1) + t_p(n)\}, \quad n = \hat{k}_1, \hat{k}_1 + 1, \ldots \tag{3.20}$$

that are initialized as $T_p(\hat{k}_1 - 1) = 0,\ \forall p$, and updated using

$$t_p(n) = \ln\left(\frac{f_{\lambda_0}(\mathbf{z}_n)}{f_{\lambda_p}(\mathbf{z}_n|\mathbf{z}_{n-1})}\right) \tag{3.21}$$

Unlike $b_p(n)$, the value of $t_p(n)$ does not depend on the value of the corresponding test statistic at time $n-1$ since conditional distributions are always used under $\mathcal{H}_1^{(p)}$ in the quiescent detection phase. The absence of any source is declared at time n whenever

$$T_{p^*}(n) \geq \gamma \tag{3.22}$$

where

$$p^* = \arg\max_p\ B_p(n) \tag{3.23}$$

and γ is the threshold for quiescent detection and p^* represents the determined class label of the detected source at time $\hat{k}_0 - 1$ i.e. the time immediately preceding the start of the newly detected quiescent period. The process then reverts back to looking for a new source of unknown type according to (3.15). This phase switching process continues indefinitely, logging the detected source each time a new quiescent period starts.

Thus, successful detection is really just the beginning step to ultimately make a correct transient classification, i.e. determining which $\mathcal{H}_1^p$ is most likely. Once the end of a source is detected (i.e. a new quiescent period is detected), the source type which has the highest test statistic $B_p(n)$ at that time is identified and the event is classified as belonging to that source type. This detection and classification scheme is illustrated in Figure 3.2. In this figure, a helicopter source signature is present in the 1/3 octave data. Along with this 1/3 octave data, the source detection test statistics as well as the quiescent detection test statistics are plotted for each observation. Since in this example the SCST algorithm was trained on two source types, there are 2 signal detection test statistics and 2 quiescent detection test statistics. This figure shows that the helicopter signal was detected at observation 60 as the source detection test statistic exceeded the source detection threshold of 100 at that time. The end of the signal was then detected at observation 103 because the quiescent detection test statistic T_p for the source that has the highest signal detection test statistic B_p has exceeded the quiescent detection threshold of 50 at this observation (and was subsequently reset to 0). Since at the end of the source signal the highest signal detection test statistic belonged to the helicopter source class, this event was correctly classified as a helicopter source.

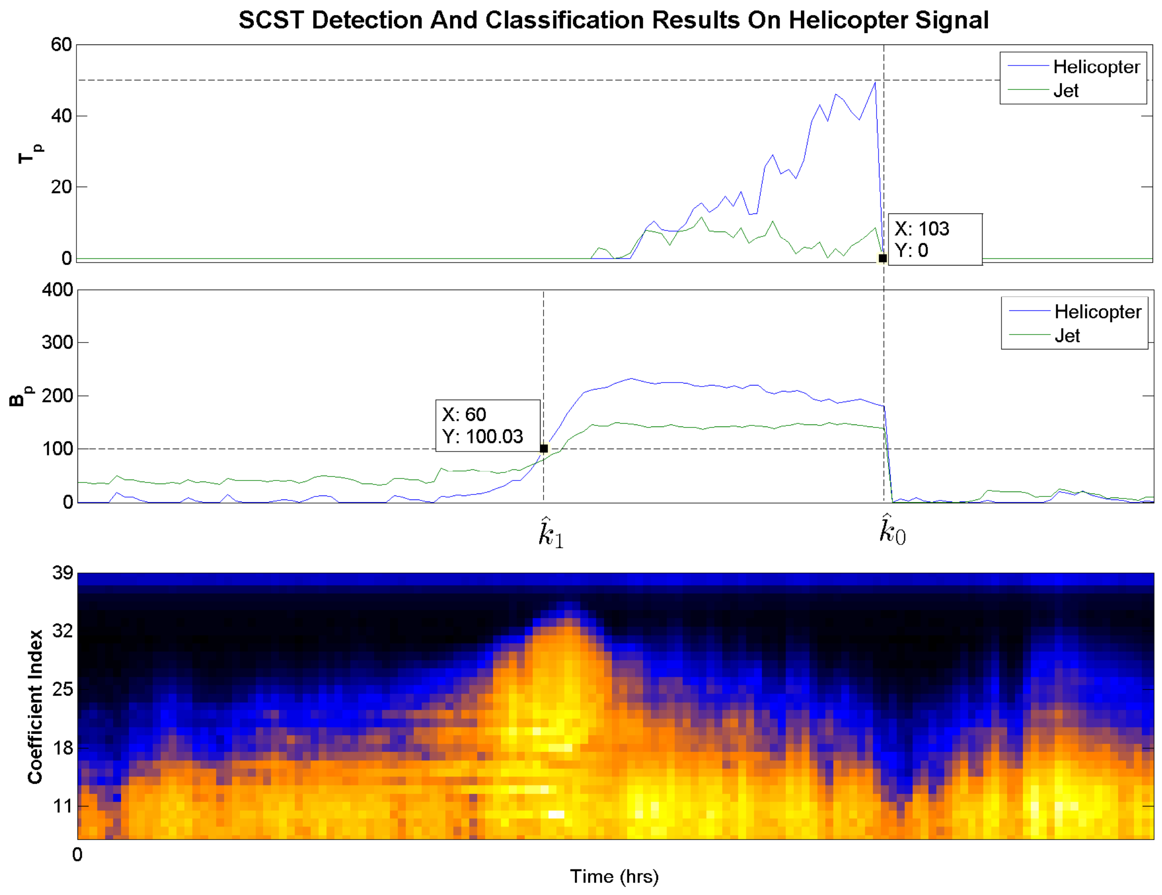


Figure 3.2: SCST Detection and Classification Example

3.2.3 Efficient SCST Implementation

Even though the above tests establish the underlying mechanisms of the SCST-based source detection and classification, we still need to compute the probability density functions $f_{\lambda_0}(\mathbf{z}_n)$ and $f_{\lambda_p}(\mathbf{z}_n|\mathbf{z}_{n-1})$ used for the updates of $B_p(n)$ and $T_p(n), \forall p \in [1, P]$ for every new quantized observation vector $\mathbf{z}_n$. This requirement is generally intractable without assuming independence of observations $\mathbf{z}_n$'s under each hypothesis $\mathcal{H}_1^{(p)}$. For this reason, the SCST algorithm uses a Bayesian network framework [19] to model the temporal evolution of a given class of source events during the training process. A Bayesian network allows for efficient calculation of a complicated joint (or conditional) probability by decomposing it into a product of conditional probabilities given other dependent states, which is much simpler to compute using the established (during training) lookup tables.

The Bayesian networks used to define a probability distribution are based on observed dependence between atoms in a set of quantized observation vectors, $\mathbf{z}_k$. First, the mutual information between coefficients in two consecutive data vectors is calculated for the entire training data set. Then, those coefficients with a mutual information above a certain user defined threshold are assumed to be dependent while the others are assumed to be independent of each other. This dependency structure is illustrated in Figure 3.3 for the third atom in a vector of size 4. The arrows indicate possible dependency of the third element on the values of elements preceding it in that vector and on the values of elements in the previous vector. These dependencies are formed based on the mutual information between the elements and so in this example it is assumed that the mutual information between all the elements connected by an arrow is above the threshold. If the mutual information between 2 elements is not high enough then there would not be an arrow between them. Although an element cannot be dependent on elements with a higher index in the same vector in order to preserve the acyclic property of the BN, this dependency can be incorporated into the BN for the higher index element. Since in the end we are interested in the conditional probability of the entire vector, all of the dependence structures for the individual vector elements will be used and so no information will be lost by assuming this dependence structure.

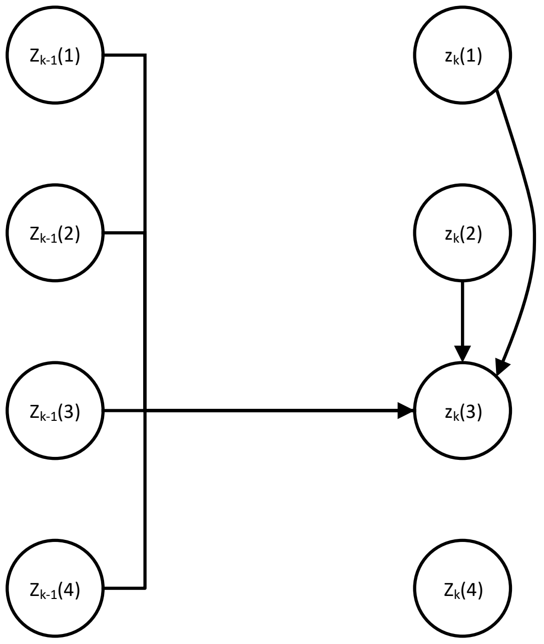


Figure 3.3: Bayesian Network Dependency Structure Example

Once this dependency structure is created, the number of times a particular atom takes on one of the L possible values is tabulated for each combination of dependencies (other atoms which this atom is dependent on). This number is then normalized by the total number of times that atom takes on any value given the particular combination of dependencies. This is done for all possible combination of dependencies to create a conditional probability density for that atom. This process is then repeated for all the other atoms in the observation vector to create a conditional probability density function for each atom. These conditional probability density functions are then multiplied together to create a single conditional probability density function for an observation vector as follows:

$$f_{\lambda_p}(\mathbf{z}_k|\mathbf{z}_{k-1},\lambda_p) = \prod_{i=0}^{S-1} f_{\lambda_p}(\mathbf{z}_k(i)|\nu_{p,i},\lambda_p) \tag{3.24}$$

where $\mathbf{z}_k \in \mathbb{R}^S$ is the quantized, sparsely coded observation vector at time k and $\nu_{p,i}$ is the state of atoms from $\mathbf{z}_k$ and $\mathbf{z}_{k-1}$ which were determined to be dependencies of atom $\mathbf{z}_k(i)$.

3.3 Conclusion

The SCST algorithm provides a computationally simple way of detecting and classifying acoustic sources in streaming 1/3 octave data. The algorithm computes probabilities of the vector belonging to one of the source types when a new data vector is captured, sparsely coded and quantized. These probabilities are then used to form likelihood ratios and aggregated over numerous observations using a CUSUM procedure. Detection of source events, quiescent events and subsequent classification of the detected sources are all done based on these accumulated ratios. The main drawback of this method is that it cannot detect multiple sources occurring at the same time. Moreover, it relies on interference signals lying in separate subspaces from those of source signals in order to separate them through sparse coding. However, even with these drawbacks, this algorithm has shown superior performance to other existing methods when detecting and classifying sources of interest over national parks.

CHAPTER 4
SCST IMPLEMENTATION ON THE EMS

4.1 Introduction

The SCST algorithm was implemented on the Artix 7 FPGA as a mix of custom logic and software running on a soft-core processor. Since the SCST algorithm was developed with the noise monitoring application in mind, it was designed to minimize the number of computations that need to be performed online. The bulk of the computations in this algorithm come from performing sparse coding for each observation because most of the other costly steps can be done offline during the training and hence don't need to be implemented on the EMS system. Once the sparse coding is done, lookup tables are used to compute the relevant test statistics for detection and classification.

In this chapter, the SCST algorithm is partitioned into several steps and the implementation of each step is discussed in detail. First, the data acquisition subsystem is mentioned and the implementation of the 1/3 octave filtering is discussed. Then, the implementation of the fast OMP algorithm used in the sparse coding step is outlined. Finally, the way the detection and classification is done on the sparsely coded vectors is discussed and the way the probabilities for each signal model are computed is explained in detail. The chapter concludes with an overview of the software/hardware partitioning that was done in the implementation of the SCST algorithm.

4.2 Implementation Overview

The entire process in SCST was first broken down into several steps as shown in Figure 4.1. Steps 1 and 2 were implemented using custom logic written in VHDL in order to meet the performance requirements of the algorithm. The rest of the steps were implemented in C code that runs on the Microblaze soft-core processor because these steps had more relaxed timing constraints and also lent themselves well to serial execution. Thus, by implementing them in software a lot of FPGA resources were saved. In addition, using the Microblaze processor for the last part of the algorithm simplified the storage and management of the detection and classification results on the SD card as there are open source C libraries that implement this functionality.

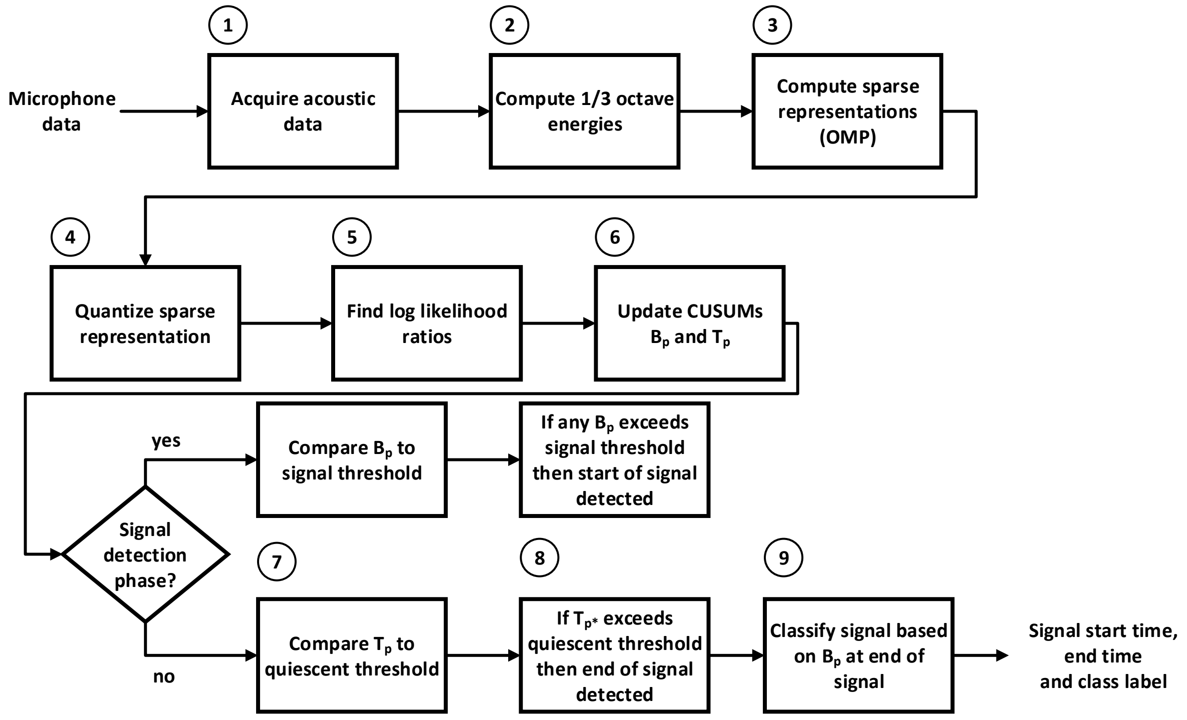

1
2
3
Microphone data
Acquire acoustic data
Compute 1/3 octave energies
Compute sparse representations (OMP)
4
5
6
Quantize sparse representation
Find log likelihood ratios
Update CUSUMs B_p and T_p
yes
Compare B_p to signal threshold
If any B_p exceeds signal threshold then start of signal detected
Signal detection phase?
7
8
9
no
Compare T_p to quiescent threshold
If T_{p*} exceeds quiescent threshold then end of signal detected
Classify signal based on B_p at end of signal
Signal start time, end time and class label

Figure 4.1: High level overview of SCST implementation

4.3 Data Acquisition

The microphone that is used to capture the acoustic signals is an InvenSense ICS-43432. As outlined in Chapter 2, this microphone outputs 24-bit digital data so no analog to digital conversion is necessary. Instead, the FPGA simply provides a clock signal to the microphone along with a few other control signals and reads in the data 1 bit at a time. This data is buffered until a full 24-bit sample is read at which point the sample is sent to the upsampling stage and the buffer is cleared. The 24-bit samples are taken at 48 kHz which is near the sampling limit of the microhpone but since the 1/3 octave filter was designed to operate on data at 96 kHz sampling rate, the data obtained by the microphone is upsampled by a factor of 2 after it is captured. This is done by placing a zero sample in between each sample and then passing the upsampled signal through a 256th order finite impulse response (FIR) low-pass filter with a cutoff frequency of 50 kHz. The output of this low-pass filter is then passed to the 1/3 Octave filter bank and the buffer is cleared to make room for the next sample.

4.4 1/3 Octave Filtering

The 1/3 octave filter reads in the filtered acoustic signal captured by the microphone one 24-bit sample at a time and outputs 33 versions of the acoustic signal each filtered with a different bandpass filter corresponding to a different 1/3 octave band (from 100 Hz to 1 kHz). The filter bank consists of 4 different digital filters. The first three filters are 8th order infinite impulse response (IIR) filters implemented using 4 second order sections with center frequencies at 0.265π, 0.333π and 0.420π as shown in the frequency response graph of the filters in Figure 4.2. These filters decompose the incoming signal into 3 1/3 octave components and meet IEC 61260 Ed. 1.0 requirements for a Class 0 device. The last filter is an 8th order IIR lowpass filter (implemented using 4 second order sections) with a cutoff frequency of 0.236π. This filter is applied to the signal after the first 3 filters and then the signal is downsampled by a factor of two. This allows the first 3 filters to be used again on the downsampled signal in order to separate out the 1/3 octave content corresponding to the next octave. The actual downsampling is done in real-time with a counter specifying which outputs of the lowpass filter to ignore (i.e. not pass to the bandpass filters). This process is repeated 11 times in order to create 33 time signals, each one corresponding to a different 1/3 octave frequency bin.

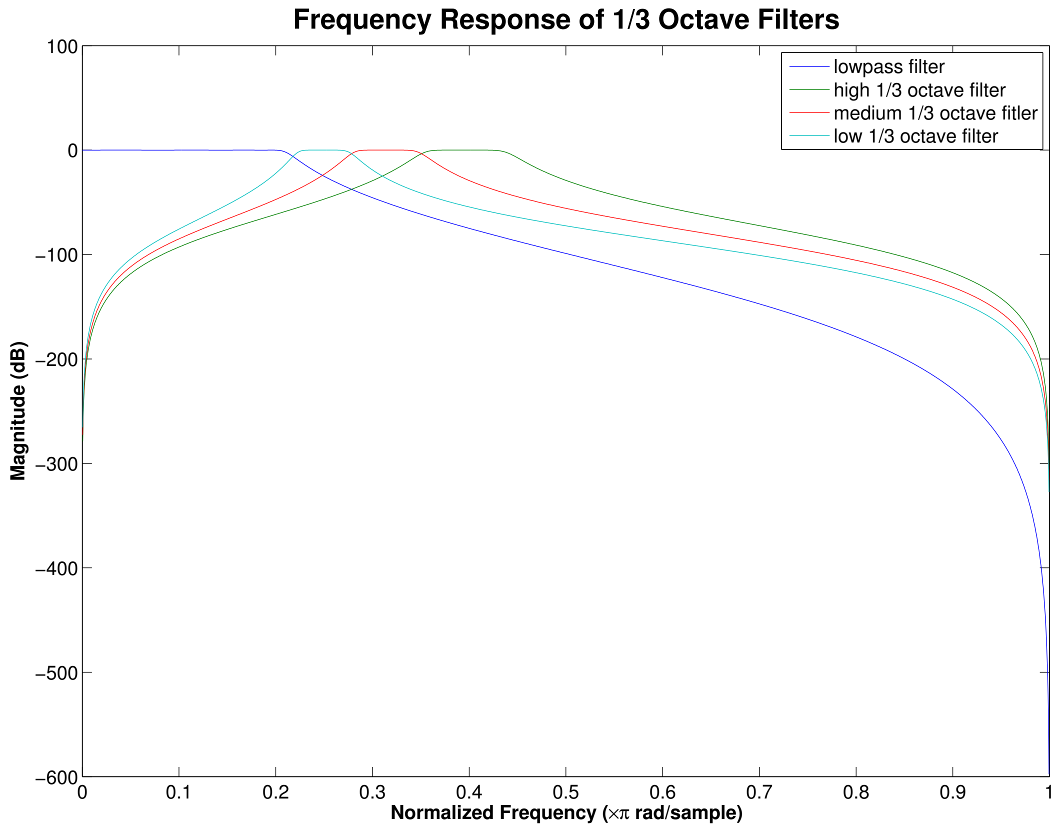


Figure 4.2: Frequency response of filters in the 1/3 octave filter bank

The power over 1 second is then computed for each of these 33 outputs. First, the outputs from all the channels are converted to double precision, squared and then added to their respective running sums as soon as they are available. At the same time, a counter keeps track of the number of clock cycles that have occurred since the last averaging step. Once a second's worth of clock cycles passes the accumulated values from all of the channels are normalized by the number of values that were summed in those respective channels. This normalization is done by a different finite state machine allowing the accumulating logic to start logging the next set of energy values right away.

Once the normalized values are calculated, they are stored in registers and an interrupt signal is sent to the Microblaze processor. When the processor receives the interrupt signal, it reads the registers using the AXI interface and retrieves these energy values. The processor then performs sparse coding, signal detection, and classification (steps 3 through 9 in Figure 4.1) and waits for the next set of energy values to be ready (indicated by the interrupt).

4.5 Sparse Coding

The sparse coding in step 3 is done using a fast OMP algorithm [39], summarized in Section 3.2.1, on the Microblaze soft-core processor instantiated in the Artix 7 FPGA fabric. This algorithm was chosen as it avoids costly operations like matrix inversion and is thus faster than the traditional OMP algorithm. The implementation of the algorithm itself is very straightforward and closely follows the procedure outlined in Section 3.2.1. First, the inner product between the 1/3 octave energy vector and the first dictionary column is calculated. This sum and the index of the dictionary column are then stored in separate variables as the maximum inner product and maximum inner product column index. Then, the inner product for the next dictionary column is computed. If that inner product happens to be larger than the one currently stored, then the maximum inner product and maximum inner product index variables are updated with this new inner product and the index of the column which produced it respectively. This is done for all of the remaining dictionary columns. Once the dictionary column which produces the maximum inner product with the 1/3 octave data vector is found, the least squares filter based on this column is computed according to (3.4). Then, the α_1 variable is computed and used to initialize the residual vector according to (3.5).

After this initialization, the entire process is repeated but with the residual vector used instead of the observation vector. As described in Section 3.2.1, the temporary variables b_{t-1}, $\bar{h}_{e_t}$, q_t and α_t are computed according to (3.7), (3.8), (3.9), and (3.10) respectively. These variables are then used to update the recursive least squares filter $\mathbf{Q}_{\tilde{\mathbf{H}}_t}$, the result vector $\hat{\mathbf{x}}_t$, and the residual vector $\mathbf{r}_t$ according to (3.12), (3.11), and (3.13) respectively. This is done at each iteration after the first until a stopping criteria is met. Also, all iterations of this algorithm after the first ignore the dictionary columns which were chosen in the previous iterations when finding the new dictionary column that has maximal inner product with the residual vector. Furthermore, each time before selecting a new dictionary column to use, the magnitude of the residual is computed and if the result is less than a user specified threshold value then the process terminates. The process also terminates after 33 dictionary columns are chosen (i.e. after 33 iterations) as that is the maximum number needed in order to represent any 33 dimensional 1/3 octave observation vector.

The dictionary utilized in Fast OMP is stored in RAM on the Microblaze as a 2-D double precision array. This is done because the entries in it are frequently referenced by the algorithm (when finding columns that have the maximum inner product with the residual) so storing them in RAM greatly increases performance. All the other computations outlined in this section are done using the floating point processing unit on the Microblaze and a double precision multiplier implemented in custom logic. The computation of the inner products as well as the iterative updating of the residual, least squares filter, and other variables used by the Fast OMP algorithm in Section 3.2.1 are all computed inside for loops. This is because all of these computations are recursive, and must be done in order, which makes loop structures ideal for their implementations. It is also why little benefit would be gained from implementing Fast OMP in custom logic.

4.6 Detection and Classification

After using Fast OMP to find the sparse codes $\mathbf{x}_k$ associated with the set of 1/3 octave energy values $\mathbf{y}_k$ that is being processed, the elements in this vector associated with interference dictionary columns are removed and then the vector is quantized element wise using pre-determined quantization levels (see Section 3.2.1) to form $\mathbf{z}_k$. The Microblaze then uses lookup tables to find the log-likelihood ratio of this quantized sparse vector having been produced by a certain source model vs the vector being simply noise. It does this by looking up the log-likelihood of each of the elements in $\mathbf{z}_k$ being from a certain source model and adding these values together. Then it looks up the log-likelihoods of each element being from the noise model and subtracts these values from the previous total. The resulting value is the CUSUM update variable $b_p(n)$ in (3.18) for that particular source model. This is done for each source model and the logs of the ratios are added to their respective running totals. If any running total B_p for a source type exceeds a pre-set threshold η then a source is detected. If B_p ever falls below 0, it is reset to 0 instead thus preventing long quiescent periods from biasing the detector and making it more difficult to detect a source.

Once a cumulative log-likelihood ratio for a particular source exceeds the threshold, a different cumulative log-likelihood ratio for each succeeding observation vector is also calculated for that source. This other log-likelihood ratio t_p describes the log-likelihood of the subsequent observations containing simply noise vs containing noise and the particular source as shown in (3.21). This value is calculated in a similar way to b_p except that the log likelihoods of each element in $\mathbf{z}_k$ being from the noise model are added together and the log likelihoods of them being from the source model are subtracted from this total. Just like for B_p, these t_p values are accumulated for each source (as shown in (3.20)) over all observations from when B_p for that source has reached the threshold η to when the end of a signal is detected and the signal is classified. Also, just like with B_p, if this cumulative ratio T_p ever falls below 0, it is reset to 0 instead.

When the ratio T_{p^*} of the source type p^* which has the highest cumulative source detection ratio B_{p^*} exceeds a pre-set threshold γ then the end of the source is detected and the source is classified as being of class p^* as shown in 3.23. After that, all the cumulative ratios (B_p and T_p) are reset to zero and the algorithm starts looking for the start of a new source signal.

The log-likelihoods of each entry in the sparse, quantized vector $\mathbf{z}_k \in \mathbb{R}^S$ are stored in a lookup table $\mathbf{U}$ where each element $u_{gpjml} = \ln \mathcal{L}(\lambda_p | z_k(j) = l)$. This lookup table has the log-likelihoods of each element in an observation vector $\mathbf{z}_k$ indexed using several variables. The first index g is a binary variable which is equal to 0 if $B_p = 0$ and is equal to 1 otherwise. The second index $p \in [0 \ldots P]$ is the class number for the particular source or noise model whose likelihood is being calculated. The next index $j \in [0 \ldots S-1]$ is the actual index of the coefficient in the quantized, sparse vector $\mathbf{z}_k$ for which the likelihood is being determined. The fourth index $m \in [0 \ldots D^{(gpj)} - 1]$ is based on the state of the dependencies of that particular coefficient in the vector $\mathbf{z}_k$ under the particular signal model. These dependencies are the atoms linked to the atom of interest by edges in the Bayesian network model for that signal class while the variable $D^{(gpj)}$ is the total number of possible combinations that can be formed with these dependencies. Finally, the last index $l \in [1 \ldots L - 1]$ is equal to the value of the quantized coefficient $z_k(j)$ for which the log-likelihood is being calculated.

In practice, this table is flattened into a 1-D array $\mathbf{v} = [v_0 \ldots v_F]$ and stored as a continuous block of memory in flash. It is stored in flash memory instead of RAM because it takes up a lot of space (over a Megabyte) and only a few entries from this table need to be read every second. Furthermore, since different entries have different number of dependencies, when this table is stored in Flash the number of dependencies of each entry under each source model is stored in an array in RAM so that the algorithm can quickly find the correct log-likelihood value for each entry. The number of dependencies is needed in order to retrieve the log-likelihood corresponding to the correct combination of those dependencies and by storing these values in RAM only a single access to flash memory is necessary in order to find the log-likelihood for a particular entry in the vector $\mathbf{z}_k$. Also, the memory blocks allocated to store the log-likelihoods for each element in $\mathbf{z}_k$ are made to be of size D^* which is the memory block size needed to store log-likelihood values for the entry if the entry had the maximum number of dependencies. The result is that for most entries which do not have very many dependencies there is some unused space in the memory which is filled with zeros. Although this is inefficient from a space standpoint, it is done in order to speed up computation because this way the algorithm does not need to check the number of dependencies of each preceding entry in the vector in order to find the offset into the memory block corresponding to a particular entry. Furthermore, the flash memory chip used on the EMS system is large enough to fit several of these log-likelihood tables so the extra memory used by this addressing scheme is not an issue. With this implementation, the elements of $\mathbf{U}$ are mapped to the elements of $\mathbf{v}$ using the relationship $v_f = u_{gpjml}$ where $f = g(P+1)SD^*L + p(SD^*L) + j(D^*L) + m(L) + l$.

4.7 Conclusion

Because the SCST algorithm has many steps which have very different computational demands and timing constraints, it was implemented partly in custom logic and partly in software running on a soft core processor. The data acquisition, 1/3 octave filtering and energy computation were all done in custom logic as these steps need to be done continuously in order to provide the SCST algorithm with observation vectors at regular intervals. The sparse coding step could also be though of as data conditioning and thus done in custom logic. However, because the adopted fast OMP algorithm is serial in nature not much performance benefit would have been gained from implementing it in custom logic while the FPGA area usage would have been substantial. Furthermore, implementing it in software allows it to be easily modified to update the dictionary matrix and also allows it to be easily replaced with another sparse coding algorithm in the future if such a need would arise. The quantization of the sparse vectors as well as the actual detection and classification were also all done in software because these steps are not very computationally intensive and mainly involve accessing a few values from lookup tables and summing them together. Thus there would be little performance benefit to implementing them in custom logic.

The final implementation of the SCST algorithm was very space efficient and only took up 22 % of the logic cells available in the Artix 7 FPGA. This allows room for other algorithms to be implemented in the future should the need arise. Even if the rest of the FPGA remains unused, the small percentage of utilization of the FPGA means that the FPGA will be drawing less power when running this algorithm and thus smaller batteries and/or solar panels can be used to power the EMS board. The implementation does make a lot of use of the RAM memory (over 50 % of the available RAM) as the Microblaze needs this memory to store the C code used to perform the sparse coding, source detection, and classification as well as to store the dictionary that is used during sparse coding and the table of dependencies for the Bayesian network models. However, this is preferable to using other FPGA resources because this same Microblaze processor can also be used to execute other algorithms (such as parts of the beamforming algorithm) when it is not busy executing the SCST algorithm. Moreover, it is better to use RAM than the other FPGA resources because more RAM can be added to later revisions of the EMS board should the need arise but adding more logic cells would require either adding another expensive FPGA chip or replacing the xc7a100tftg256-2 FPGA with a bigger (and more expensive) chip. As such, this implementation of the SCST algorithm is both flexible and space efficient while still being able to perform detection and classification of acoustic sources every second in near-real time.

CHAPTER 5
SCST IMPLEMENTATION VALIDATION

5.1 Introduction

To validate the SCST implementation explained in Chapter 4, the accuracy of each of the stages in the algorithm were first checked by performing modular tests on the various subsystems. First, the combined frequency response of the microphone and 1/3 octave filter on the EMS system were compared to that of the industry standard Larson Davis 831 noise level meter. Then, the accuracy of the 1/3 octave filter and OMP implementations were separately compared to their implementations in MATLAB in order to ascertain any performance differences between the two platforms. After that, the implementation of the detection and classification portions of the SCST algorithm was evaluated on a 2 hour segment of previously collected acoustic data. Finally, a full system test (including training the KSVD dictionaries and BN parameters) was conducted at Lake Mead to evaluate the performance of this system.

In this chapter, each of these validation tests are presented in the order in which they were conducted and the results are discussed. The chapter then concludes with a summary of the evaluation results and a discussion of the benefits and drawbacks of this implementation.

5.2 Microphone Response

In order to verify the data acquisition system, the data gathered by the EMS system was compared to those gathered by the Larson Davis meter. Because the Larson Davis system can not record raw acoustic data and the EMS system also lacks the memory bandwidth needed to do this, the microphones used by the two systems could not be compared directly. As such, the two systems were configured to record 1/3 octave energy levels instead. These two systems were then set up side-by-side in an anechoic chamber and pure tones were played from a speaker 5 feet away from the two systems. The tones were played at center frequencies of 20 1/3 octave bins (starting from 250 Hz and going all the way to 20kHz). The center frequencies of the Larson Davis were used for this test which were slightly different than the center frequencies of the EMS board. This is because the Larson Davis uses the more convenient, non-fractional center frequencies (e.g. 400 instead of 396.9) outlined in ISO 3:1973 standards which are not always exactly 1/3 of an octave away from the center frequencies of adjacent bands. The EMS 1/3 octave filter bank on the other hand always uses the exact 1/3 octave center frequencies as it is implemented using only 4 different filters and downsampling. Also, only the top 20 1/3 octave bins were tested out of the 33 1/3 octave bins that are calculated by each system because the speaker that was used for testing could not produce tones below 250 Hz. Each tone was played at 2 different amplitudes and the difference between the energies reported by the EMS system and those reported by the Larson Davis system were computed and plotted for each of the tested frequency bins and for each of the two amplitude levels as seen in Figure 5.1.

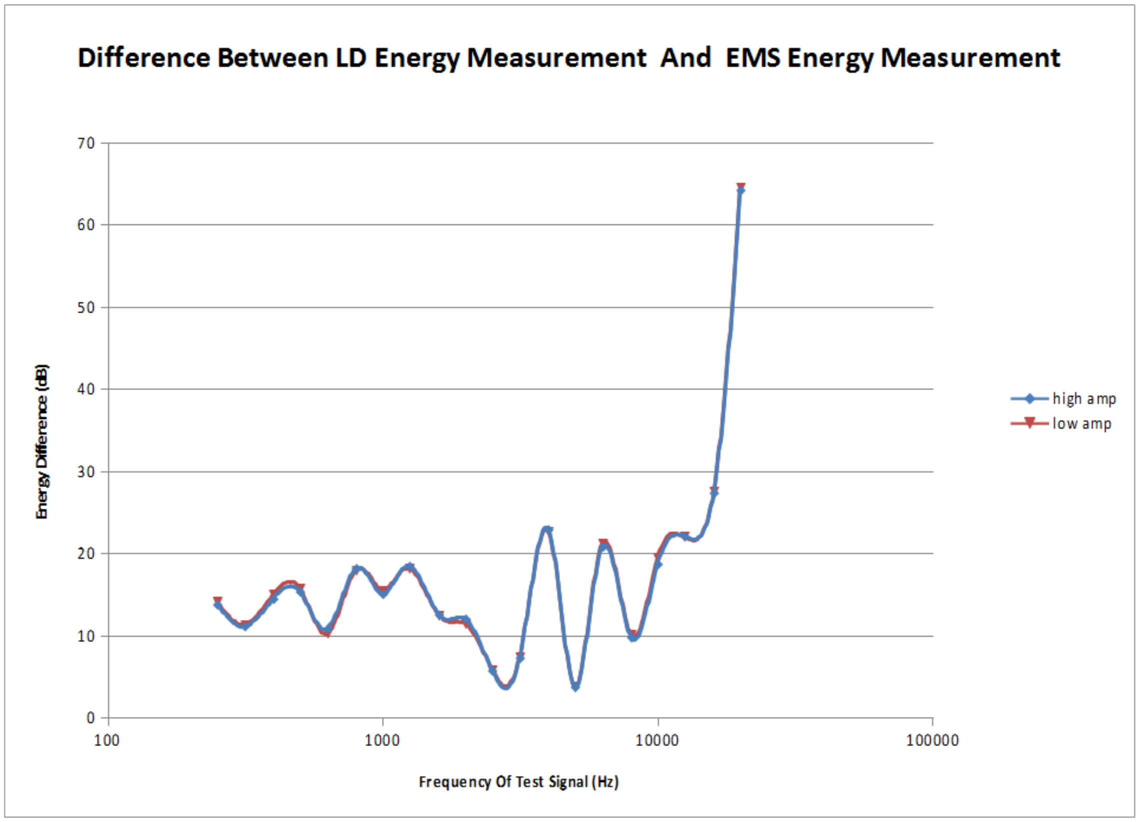


Figure 5.1: Microphone comparison graph

As can be seen from Figure 5.1, the combined microphone and 1/3 octave filter frequency responses of the two systems had quite different characteristics. First of all, the EMS system seemed to have a more sensitive microphone as the energy levels for the test signals reported by the EMS system were in general higher than those reported by the Larson Davis which is why the energy difference between the two systems was always positive. Moreover, the EMS system appeared to have a higher gain at higher frequencies as the differences between the energy readings reported by the two systems were much higher for the frequency bins above 10 kHz. The first discrepancy could be attributed to a mis-calibration and can easily be remedied by subtracting a constant value from the energies computed by the EMS before they are reported. However, the higher gain at high frequencies exhibited by the EMS is likely due to some non-linearity in the frequency response of the microphone used for this board as the frequency response of the microphone used on the EMS board also steeply increases past 10 kHz as reported in its datasheet. However, since this non-linearity appears only after 10 kHz and our sources of interest lie between 100 Hz and 1 kHz, this discrepancy between the two systems could probably be ignored. The differences in the frequency responses of the two systems at lower frequencies can be attributed to many different factors including the differences in the acoustic properties of the microphone housings, the aforementioned differences in center frequencies of some of the bins that were tested, and the differences in the frequency responses of the 1/3 octave filters used by the two systems.

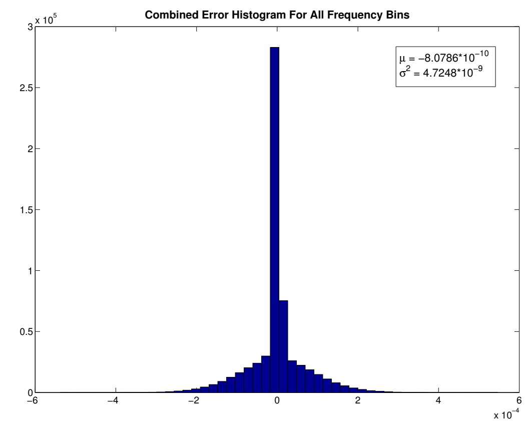
The one conclusion that can be drawn from this test is that the combined frequency response of the 1/3 octave filters and microphones for the two systems are significantly different and it would be impractical to try and create a compensation filter on the EMS system in order to have it display the same frequency response as the Larson Davis. This means that the data captured by the Larson Davis cannot be used to train the EMS system and instead new data will need to be collected using this EMS system for training the SCST algorithm before the system is actually deployed to perform signal detection, classification and tracking at a specific location.

5.3 1/3 Octave Filter Accuracy

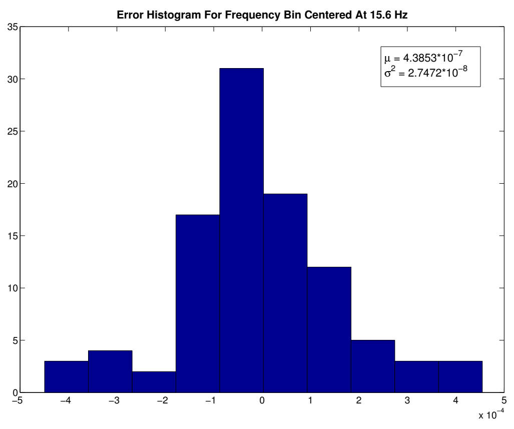
In order to verify the accuracy of the implementation of the 1/3 octave filter a 1 second linear chirp signal (going from 0 Hz to 20 kHz) was generated in MATLAB and passed through the MATLAB implementation of the 1/3 Octave filter bank. Then, the same signal was passed through the functional simulation of the FPGA implementation of the filter bank. The differences between the outputs of the two implementations for each of the 33 frequency bins were computed and plotted as histograms. The histogram for the combined errors between all the frequency bins as well as the histograms for 5 select frequency bins are presented along with their respective mean and variance values in Figure 5.3.

From these tests, it was found that the maximum discrepancy (in magnitude) between the outputs of the two implementations was only $5.4682 * 10^{-4}$ (while the amplitude of the input signal was between -1 and 1) and the distribution of this error appeared to be Gaussian, as seen in Figure 5.3, with a mean of $-8.0786 * 10^{-10}$ and a variance of $4.7248 * 10^{-9}$. Furthermore, the errors in each of the 1/3 Octave bins were also Gaussian with a mean very close to zero and the variance slightly higher at lower frequencies as is shown in Figure 5.3 for select low, medium, and high frequency bins.

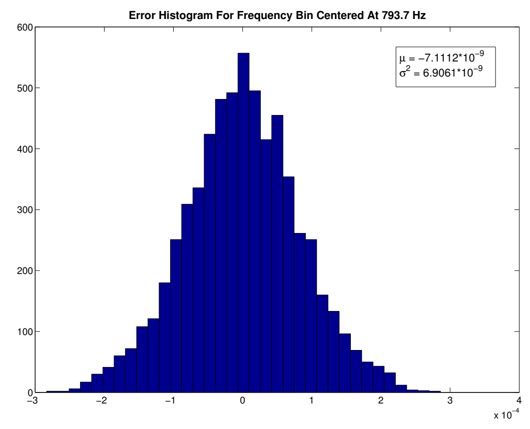
These errors can be attributed to rounding in the FPGA implementation of the filter bank as it is implemented using fixed point arithmetic on the FPGA while the MATLAB implementation uses 64 bit double floating point precision. The Gaussian distribution of the errors is likely due to the cumulative effects of rounding occurring at every operation in the filtering process as according to the Central Limit Theorem, the sum of multiple random variables (like rounding errors) approaches a Gaussian distribution as more of them are summed together. The increased variance at lower frequency bins is likely due to the fact that outputs of lower frequency bins are filtered more than those of higher frequency bins so the cumulative rounding effects are greater at lower frequencies. However, even at low frequencies, this source of error in the FPGA-based filter bank is not significant enough to impact the performance of the subsequent stages of the algorithm as such minor changes in the energies of an observation vector would not significantly impact the sparse representation of that vector and would be completely eliminated once that sparse representation is quantized.



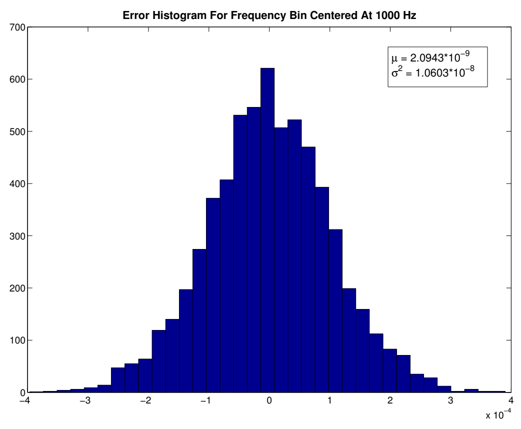
(a) Combined error histogram



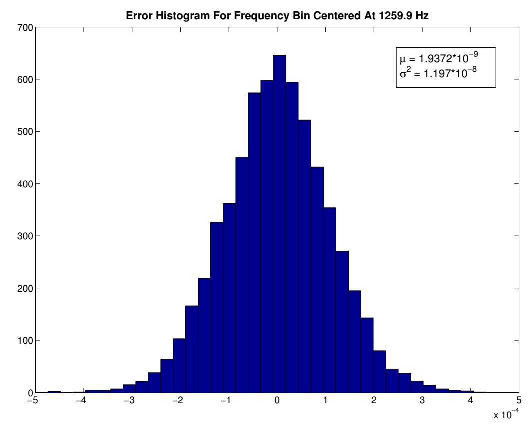
(b) 15.6 Hz bin error histogram



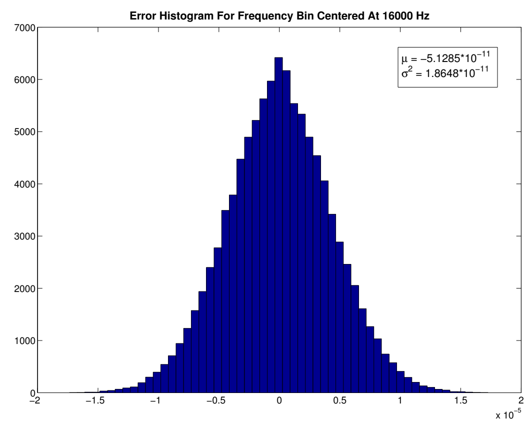
(c) 793.7 Hz bin error histogram



(d) 1000 Hz bin error histogram



(e) 1259.9 Hz bin error histogram



(f) 16000 Hz bin error histogram

Figure 5.3: Error histograms for EMS 1/3 octave filter implementation

5.4 Fast OMP Accuracy

The Fast OMP step was also tested by passing a pre-recorded 1/3 octave energy data sequence through the FPGA implementation of the algorithm as well as the MATLAB implementation. This sequence was captured using a Larson Davis meter at the Great Sand Dunes National Park and was stored on the flash memory of the EMS for testing the FPGA implementation of the Fast OMP algorithm. The dictionary used for these tests was obtained by using the KSVD algorithm in MATLAB on a training data set that was captured by the same Larson Davis meter at the same location in the national park. This dictionary matrix was stored in the block RAM memory of the EMS board for the FPGA portion of the tests. The sparse vector sequences produced by the two implementations were then compared to see whether the same atoms were chosen for signal representation for each of the input 1/3 octave vectors and also to see how close the values of the non zero elements were to each other between the two implementations.

The double precision implementation of the Fast OMP algorithm on the FPGA produced comparable results to the MATLAB implementation (which also uses double precision). The log error image for the first 224 seconds of this data set which represents the log of the magnitude of the difference in the sparse representations produced by the two implementations is shown in Figure 5.4. The maximum difference between the atom weights in the two representations did not exceed $5.2302 * 10^{-7}$ (in magnitude) and most of the atoms had no error at all as seen in the log error image. Furthermore, the algorithms selected the exact same dictionary atoms for reconstruction of all the 1/3 octave vectors that were being tested. Because of this, after quantization (with $L = 4$), the vector sequences produced by MATLAB and EMS were identical. This is important because the training of the Bayesian networks and OMP dictionary matrix are done offline in MATLAB and if the MATLAB implementation were to produce different sparse, quantized observation sequences than the EMS for the same data then these parameters might not work as well when detecting and classifying events based on sparse, quantized observations computed by the EMS. However, since the two implementations produced identical quantized observation sequences, the double precision implementation of the Fast OMP algorithm on the FPGA was deemed to be sufficiently accurate for use in the SCST algorithm.

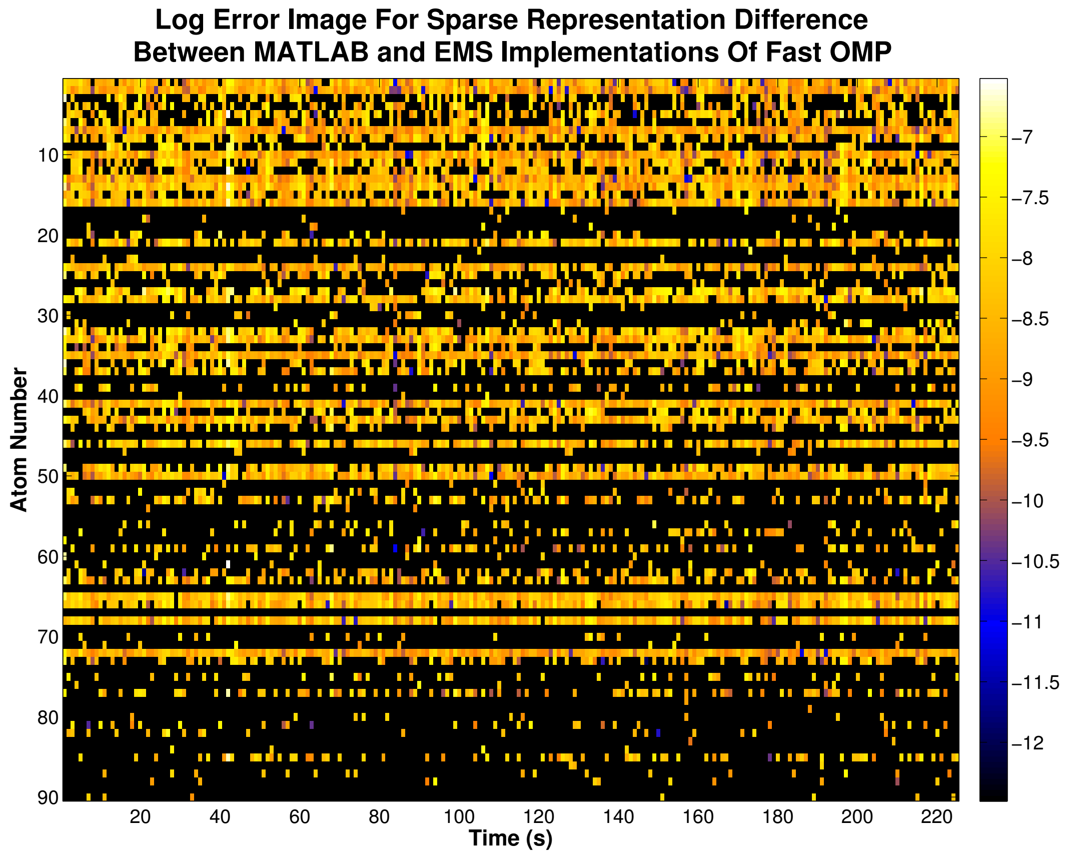


Figure 5.4: Log error image for first 224 seconds of fast OMP validation test

5.5 Full System Tests

5.5.1 System Test On Pre-collected Datasets

After verifying the individual steps in the SCST algorithm, two full system tests were conducted. First the system was tested on pre-recorded data. To perform this test, the KSVD dictionaries and BNs (described in Section 3.2.1) were formed for all the signals based on a training dataset consisting of several days of data from the Grand Sand Dunes National Park. These parameters were then uploaded to the EMS system along with 2 hours of testing data from the same park. The system was then tested with the data being read from Flash memory instead of from the 1/3 octave filter. The classification results along with the log-likelihood ratios for signal and quiescent detection computed by the algorithm were recorded for this test data and compared to the associated values and classification results produced by MATLAB for that same data set.

Figure 5.5 shows the signal detection log-likelihood ratios (LLR) B_p (computed according to (3.17)) computed by both SCST implementations. This figure shows that the B_p values computed by the two implementations were almost identical for both of the sources present in the data set. Similarly, Figure 5.6 shows the quiescent detection LLR T_P (computed according to (3.20)) as a function of time for both signals and both implementations. Just like with the B_p values, the T_p values were also almost identical for the two implementations. Since the previous validation test in Section 5.4 showed that the two implementations produced identical quantized, sparse representations for the same 1/3 octave energy data, these figures show that the LLR computations performed by the EMS implementation of SCST are consistent with the computations performed by the MATLAB implementation.

Figure 5.7 shows the detection and classification performance of the two SCST implementations on this data set. The top image in the figure is the raw 1/3 octave data that was used by both SCST implementations while the bottom two images use colored strips to indicate when each implementation detected any sources and the class label that they assigned to those sources. Although both implementations missed several source signals that occurred before the 1000 second mark, they both reported identical detection and classification results as seen by the locations of the colored strips in the bottom two images.

Since these LLRs are the test statistics used by the SCST algorithm to perform detection and classification, it is not surprising that the detection and classification performances of the two implementations were identical given the nearly identical B_p and T_p values reported by both implementations. This implies that although the microphone and the 1/3 octave filter responses on the EMS system differ from those of the Larson Davis (see Section 5.2), the actual SCST algorithm implemented on the board performs exactly as it does in MATLAB.

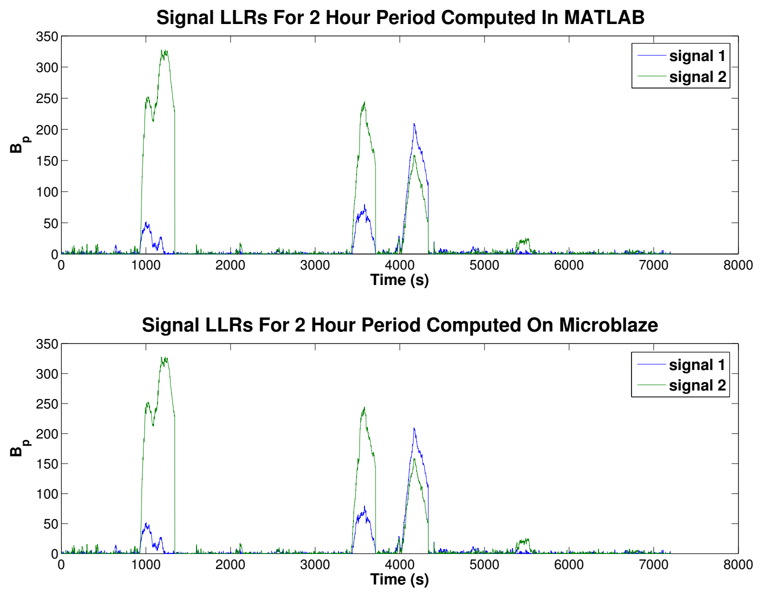


Figure 5.5: Signal detection LLR comparison graph

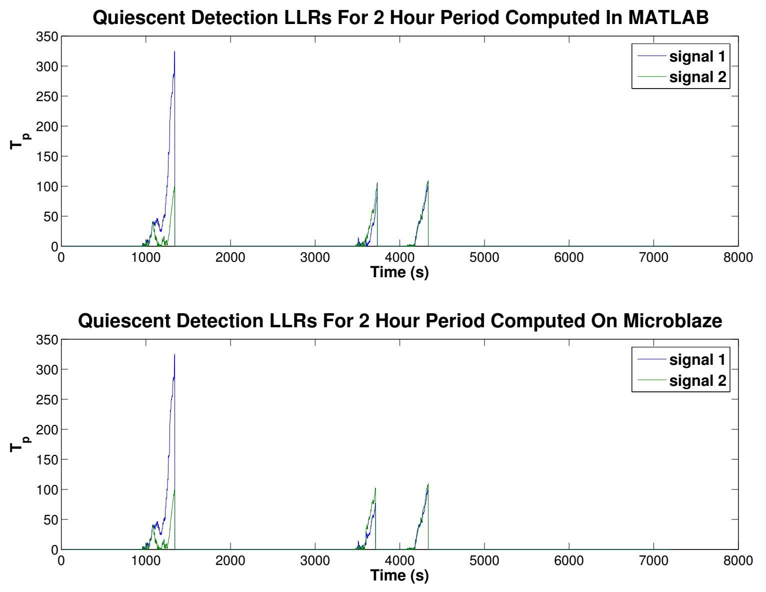


Figure 5.6: Quiescent detection LLR comparison graph

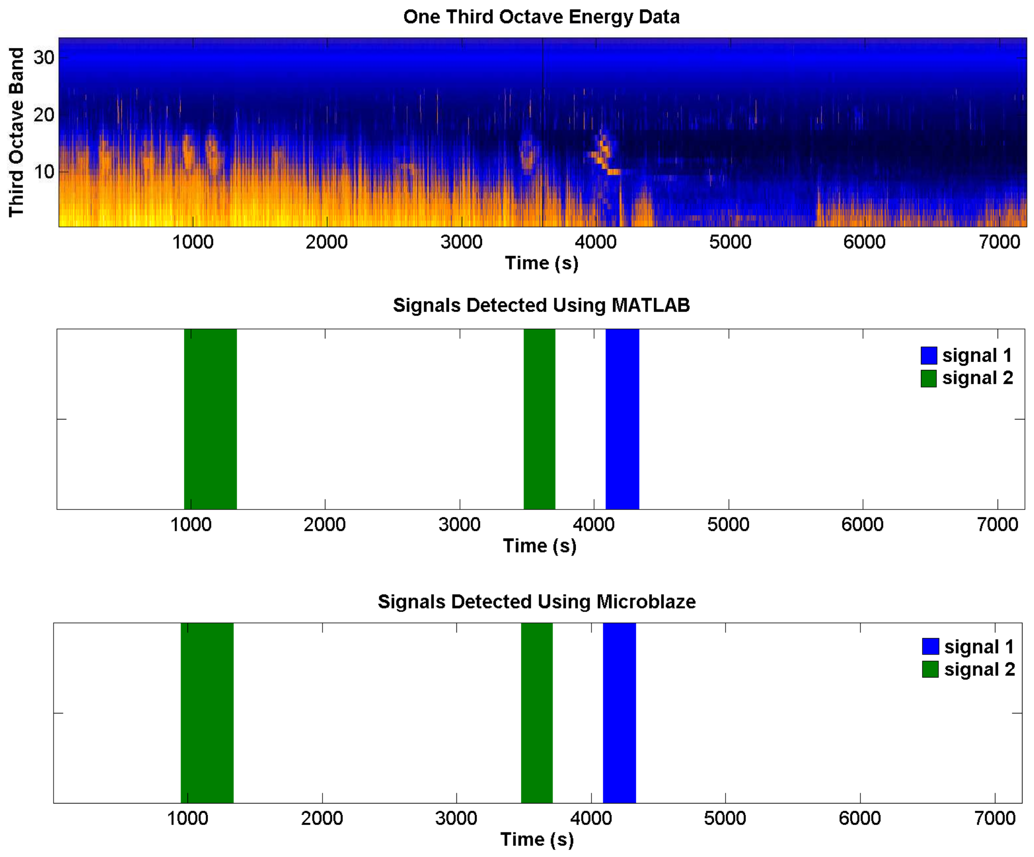


Figure 5.7: Comparison of detected events

5.5.2 System Test On Actual Field Deployed System Data

After these laboratory tests, the entire FPGA implementation of the SCST algorithm was tested on an actual field-deployed system at a national park. First, the EMS system was sent out to Lake Mead National Recreation Area for 48 hours (from 7/5/206 to 7/7/2016) to collect acoustic 1/3 octave energy data that was used for training purposes. Once the data was collected, it was manually annotated to identify and mark sources of interest. This data was then used to train the algorithm by finding KSVD dictionaries for each signal type and log-likelihood values for each BN source model (see Section 3.2.1). These dictionaries and log-likelihood values were then stored in the EMS and it was reconfigured to perform detection and classification using these parameters. The trained system was then sent back out to Lake Mead for another 48 hours (from 8/18/2016 to 8/20/2016) to perform the actual field test. There, it captured 1/3 octave streaming data and performed in-situ detection and classification on that data.

The system was trained to detect and classify helicopters and jets as these were the only sources of interest present at this location. The helicopter signals were much more prevalent in this location when compared to jets as it is a high-traffic area for air tours. Not only that but the helicopter signals were also of much higher intensity than the jet signals and thus easier to identify even in windy conditions. This is likely because the helicopters there usually fly at lower elevations as part of the tour while the jets all fly at high altitude.

The only prevalent interference at this location happened to be the wind as there were no other natural sounds (either animal or weather related) present during the time that the system was set up at the park. The wind however did vary in intensity throughout the test and even though the microphone used to capture the acoustic data on the EMS board had a wind screen, the wind would at times be strong enough so as to saturate the microphone input to the point of drowning out other signals and producing observation vectors with high energy content in all the 1/3 octave frequency bins. However, most of the times the wind was fairly moderate and its 1/3 octave signature only occupied the frequency bins below 100 Hz. Through validation testing on parts of the training data set it was determined that training on this type of wind produced better detection and classification results than when training on the intense wind as well. Therefore, for the actual field test, moderate wind was considered as the only interference class.

As seen in the confusion matrix shown in Table 5.1, the EMS was able to correctly classify the majority of the 271 helicopter sources that were detected during the field test. It did have more trouble correctly classifying the 73 jet sources that were detected during the test and also falsely detected 135 jet signals when there were no sources present in the data. This is likely due to the training data that was used for the jet source class. Since there were much fewer jet sources than helicopter sources in the training data set we had to use jet sources that were weak in amplitude and/or contaminated by wind for training. As a result, the BN model learned for this source type was likely to be less accurate than the BN model learned for the helicopter class resulting in poor generalization to the test data. Furthermore, since these jet sources are in general quite a bit weaker than the helicopter sources, it is much harder to distinguish them from the interference and so the classification accuracy might have suffered as a result.

Table 5.1: SCST Confusion Matrix for Indian Pass Data

Truth \ Decision	helicopter	jet
none	$\frac{73}{208}$	$\frac{135}{208}$
helicopter	$\frac{237}{271}$	$\frac{34}{271}$
jet	$\frac{25}{73}$	$\frac{48}{73}$

The receiver operating characteristic (ROC) curve for this detector (Figure 5.8) was created based on the class of individual observations. That is to say, the percent detection for each point on the curve was calculated as the number of observations that were correctly determined by SCST to have contained a source class divided by the total number of observations which contained a source class based on the ground truth data. Likewise, percent false alarm was calculated as the number of observations which were incorrectly determined by SCST to have contained a source class divided by the number of observations which did not contain a source class based on the ground truth data. The reason the ROC curve was generated this way is because since the sources can span various number of observations, as the detection threshold is decreased the number of false detections does not necessarily change monotonically. As new false detections appear in the data, old ones might combine together due to the observations between them exceeding the detection threshold. Therefore, it would be difficult to accurately calculate a meaningful percent false alarm rate based on the number of falsely detected sources.

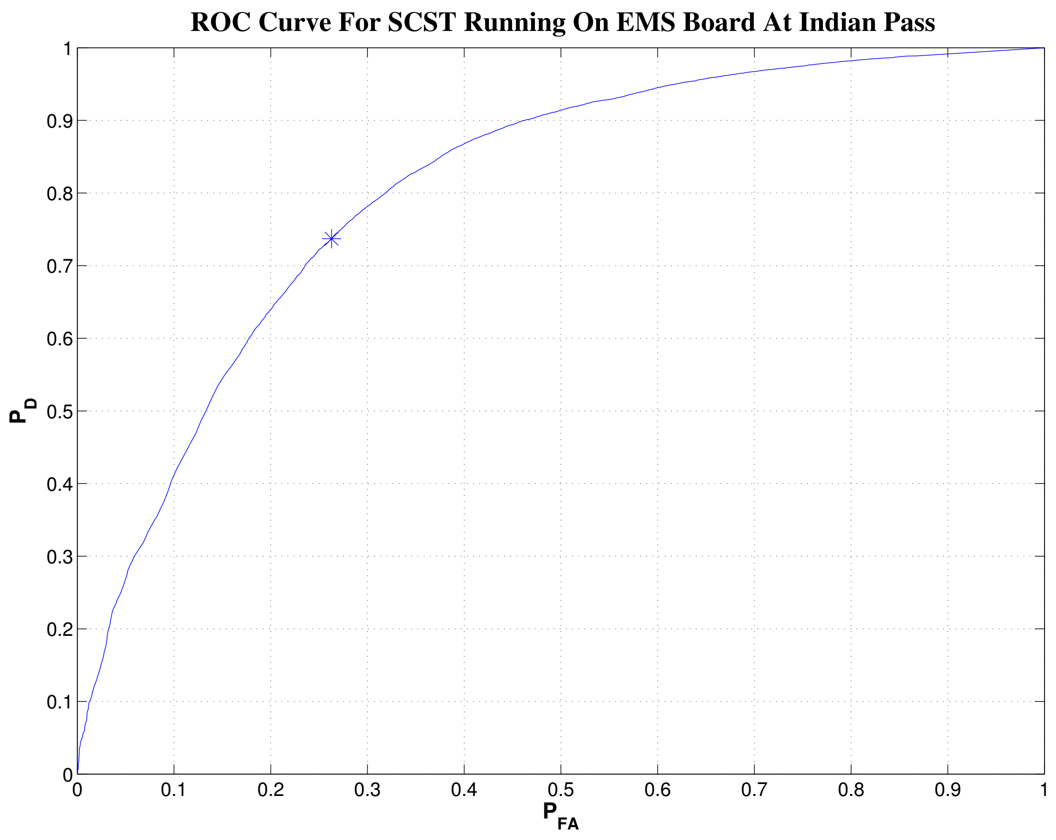


Figure 5.8: SCST ROC Curve For Indian Pass Data

This ROC curve shows that the EMS implementation of SCST performed markedly worse on this data set then the MATLAB version of this algorithm did on the Great Sand Dunes and Kenai Fjord data sets [22]. The knee point of the ROC at which $P_{FA} + P_D = 1$ was at 73.7% detection and 26.3% false alarm. Most of the false alarms occurred during periods of intense wind where the microphone was saturated and the 1/3 octave filter reported high energy content in all frequency bins as seen in Figure 5.9. This is expected because while the energy in the lower frequency bins would likely be attributed to the wind and removed by sparse coding, the energy present in the higher frequency bins would most certainly look much more like some signal model (where there is energy present in the higher frequency bins) rather than the noise model (where energy is distributed about evenly between all frequency bins). This would result in the LLR variable B_p increasing during these periods of very strong wind eventually resulting in detections where there are no sources present.

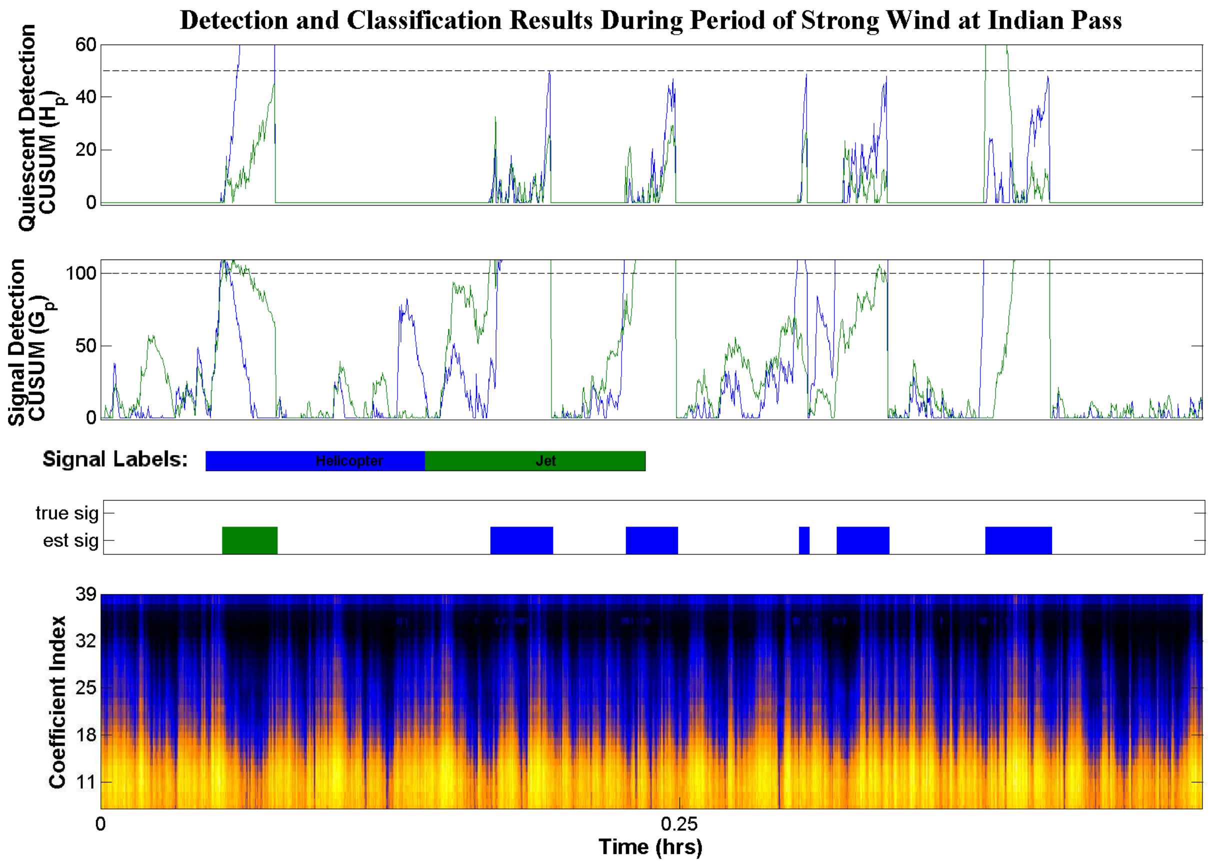


Figure 5.9: Detection and Classification Results During Strong Wind at Indian Pass

Also, as previously mentioned, many of the jet sources recorded at this location are of low amplitudes and often obscured by interference. Because of this, they are often either misclassified by the algorithm or not detected all together. This is likely another reason why the detection and classification performance of SCST on this data set was worse than its performance on the Great Sand Dunes and Kenai Fjords data sets reported in [22]. Therefore, if better training data is collected and the various SCST parameters are fine tuned, it is likely that the performance of the EMS at this location can be improved.

5.6 Conclusion

In this chapter, the results of the implementation of SCST algorithm on an Artix 7 FPGA-based EMS board were presented. Each of the major subsystems associated with data conditioning, fast OMP, and the SCST algorithm itself were tested independently. The Microphone and 1/3 octave filter on the EMS system were tested against the Larson Davis meter and determined to have significantly different performance characteristics. This was not surprising considering the vast differences in signal quality between the Larson Davis microphone and filtering system vs the inexpensive InvenSense ICS-43432 microphone that was used on the EMS. While the frequency response of the EMS system microphone and filter can be aligned with that of the Larson Davis by constructing an inverse filter based on the difference in the frequency characteristics of the two systems, such a filter will use up a significant amount of FPGA resources because it will need to be a fairly high order filter to compensate for the behavior seen in Figure 5.1. Moreover, the SCST algorithm performance should not be adversely affected by using the 1/3 octave data output by the EMS without a compensation filter assuming that the algorithm was also trained on the data captured by the same system.

The other steps in the SCST algorithm that were implemented on the EMS were shown to be comparable to their MATLAB counterparts with only slight differences that can be attributed to precision differences between the two implementations. Furthermore, all the steps in the algorithm after 1/3 octave filtering and energy computation were tested together on acoustic data captured by the Larson Davis and the detection and classification results of the EMS and MATLAB implementations were identical in terms of the start and end times of detected sources as well as the class of those sources. This shows that the EMS implementation of the SCST algorithm performs just as well as the MATLAB implementation.

Lastly, the EMS implementation of SCST was tested in an actual field-test at the Lake Mead Recreational Area. This test revealed that although there is still room for improvement of the algorithm, the EMS system performed well given the strong wind that is often present at that particular deployment location and the lack of robust training data. More importantly, the test showed that the EMS board is able to reasonably perform detection and classification of acoustic sources in near real-time. More tests should be conducted in order to better evaluate the performance of the system and it's limitations.

CHAPTER 6
SOURCE TRACK FORMATION USING WIDEBAND CAPON BEAMFORMING

6.1 Introduction

Forming tracks of the detected and classified sources is also an important function of the EMS system because with this information NPS will be able to more easily identify which areas of a national park are mostly affected by aircraft noise. In order to effectively form the trajectory or track of an object based on it's acoustic signature some kind of beamforming needs to be used. Since the sources of interest in our application are broadband (around 100Hz to 1kHz), the beamforming algorithm needs to take that into consideration. For this application, we chose a wideband extension of the Capon beamformer [26] for direction of arrival (DOA) estimation. This beamformer is relatively simple to implement yet has a narrow enough beam pattern that yields good precision DOA estimates for our application. The wideband Capon beamformer is based on incoherent combining of narrowband Capon beamformers for frequencies which the source signature occupies. As such, it consists of simply finding the power spectra of all of these narrowband Capon beamformers and combining them into a single power spectrum using arithmetic or geometric averaging.

In order to use this algorithm for track formation the beamforming algorithm needs to be run frequently enough to obtain a representative set of DOA estimates from which the trajectory of the source can be interpolated. Since the sources of interest (planes and helicopters) move quite quickly and their acoustic signatures are often only audible for half a minute to a minute, the beamforming algorithm was configured to produce a DOA estimate every second. This criteria puts a significant timing constraint on the algorithm and is the main reason that the algorithm was implemented primarily in custom logic with only the power calculations and maximization being done in software as these steps would not benefit much from the pipelining and parallelisation offered by a custom logic implementation. As previously mentioned, in order to simplify testing, the wideband Capon beamforming algorithm was implemented to continuously output DOA estimates as if a known source is always present. Therefore, when it is combined with the SCST detection and classification algorithm, it will need to be modified to buffer data while a source is present and then only perform DOA estimation on that buffered data when the source type is identified.

The wideband Capon beamforming algorithm implementated on the EMS system was tested in the field by flying an electric model plane near the array at a known trajectory and using the beamforming system to form DOA tracks for this plane. These tracks were compared to the known trajectory of the plane to determine whether this algorithm could correctly identify the heading of the plane relative to the array.

In this chapter we first review the wideband extension of the Capon beamformer. After that, the implementation details of this algorithm are discussed and the partitioning between custom logic and software is explained. Next, the test results of the algorithm when tracking the model plane are presented. Finally, this chapter is concluded with an overview of the beamforming algorithm implementation and a discussion of the results of the field test.

6.2 Wideband Capon Beamforming - A Review

The wideband Capon beamforming method [26] is based on breaking down a signal into multiple narrowband components using FFT, finding the Capon power spectrum at each narrowband component, and then averaging these power spectra together to form a power spectrum for the whole signal. This aggregated power spectrum can then be used to estimate the DOA of the source. As the name implies, these narrowband power spectra are calculated using the standard Capon beamformer [42].

The wideband Capon beamformer assumes that the source is stationary over the observation snapshot (1 second) for which the DOA estimate is to be computed. This way, the frequency domain representation of the source captured by the microphone array can be written as

$$\mathbf{x}(\omega,\theta,\phi) = \mathbf{v}(\omega,\theta,\phi)F(\omega) + \mathbf{n}(\omega) \tag{6.1}$$

where $\mathbf{x}(.)$ is the Fourier Transform of the source signal, $\mathbf{n}(\omega) \in \mathbb{C}^N$ is zero mean white Gaussian noise, and $\mathbf{v}(\omega,\theta,\phi) \in \mathbb{C}^N$ is the array steering vector which compensates for the time delay between the N different microphones. The narrowband component of the signal $F(\omega)$ at frequency ω is assumed to come at an azimuth angle θ and elevation angle ϕ relative to the microphone array. For our 5 microphone wagon-wheel array the steering vector used for the tests $\mathbf{v}(\omega,\theta,\phi) = [e^{-j(\omega r/c)\sin\theta}, e^{-j(\omega r/c)\cos\theta}, 1, e^{j(\omega r/c)\sin\theta}, e^{j(\omega r/c)\cos\theta}]$ where $r = 0.098$ was the radius of the wagon wheel array, and $c = 344$ is the speed of sound in air. This vector assumes that the microphone inputs are ordered as 1:East, 2:South, 3:Center, 4:West and 5:North in the observation vector $\mathbf{x}(\omega,\theta,\phi)$. The vector was not dependent on ϕ because elevation angle was not estimated in our tests and instead was set at 90 degrees. This was done because knowing the elevation of an aircraft over a national park is not currently useful for NPS and not having to find this angle drastically reduces the computational complexity of DOA estimation.

Once a source is detected and classified, the beamforming algorithm starts reading data from all the microphones on the array at a 48 kHz sampling rate. After collecting a second's worth of data, it breaks the data from each microphone into K non-overlapping segments of length 1024 and performs FFT on these segments. The transformed observation vector for the k^{th} segment at narrowband component ω is denoted by $\mathbf{x}_k(\omega)$, $k = 1, 2, ..., K$. These narrowband components are assumed to be independent and identically distributed (i.i.d) between different frequency bins and are used to approximate the narrowband covariance matrices by averaging the rank one spatial narrowband covariances of individual segments as follows:

$$\mathbf{R}_{\mathbf{xx}}(\omega) = \frac{1}{K}\sum_{k=1}^{K}\mathbf{x}_k(\omega)\mathbf{x}_k^H(\omega) \tag{6.2}$$

The Capon beamformer $\mathbf{w}(\omega,\theta,\phi)$ is a linear filter that tries to recover the original narrowband signal components $F(\omega)$ from $\mathbf{x}(\omega)$ by finding the minimum variance unbiased estimate of $\mathbf{x}(\omega)$ for a given set of incidence angles θ and ϕ. It can be shown [42] that this leads to

$$\mathbf{w}^H(\omega,\theta,\phi) = \frac{\mathbf{v}^H(\omega,\theta,\phi)\mathbf{R}_{xx}^{-1}(\omega)}{\mathbf{v}^H(\omega,\theta,\phi)\mathbf{R}_{xx}^{-1}(\omega)\mathbf{v}(\omega,\theta,\phi)} \tag{6.3}$$

To determine the actual incidence angles of the source, the power spectrum of the beamformed signal $\mathbf{w}^H(\omega,\theta,\phi)\mathbf{x}(\omega)$ is analyzed and the angles at which the power is maximum are found. This power spectrum of this beamformer can be computed using

$$\begin{aligned} q(\omega,\theta,\phi) = E\left[(Y(\omega))^2\right] &= \mathbf{w}^H(\omega,\theta,\phi)\mathbf{R}_{xx}(\omega)\mathbf{w}(\omega,\theta,\phi) \\ &= \frac{\mathbf{v}^H(\omega,\theta,\phi)\mathbf{R}_{xx}^{-1}(\omega)\mathbf{R}_{xx}(\omega)\mathbf{R}_{xx}^{-1}(\omega)\mathbf{v}^H(\omega,\theta,\phi)}{(\mathbf{v}^H(\omega,\theta,\phi)\mathbf{R}_{xx}^{-1}(\omega)\mathbf{v}(\omega,\theta,\phi))(\mathbf{v}^H(\omega,\theta,\phi)\mathbf{R}_{xx}^{-1}(\omega)\mathbf{v}(\omega,\theta,\phi))} \\ &= \frac{1}{\mathbf{v}^H(\omega,\theta,\phi)\mathbf{R}_{xx}^{-1}(\omega)\mathbf{v}(\omega,\theta,\phi)} \end{aligned} \tag{6.4}$$

Since this is a narrowband power spectrum and our sources of interest are wideband, the wideband Capon beamformer [26] finds these power spectra for each frequency of interest and then arithmetically or geometrically averages them to produce a single, incoherently averaged power spectrum. The geometrically averaged wideband Capon beamformer spectrum used here is

$$\mathbf{Q}_G(\theta,\phi) = \prod_{j=1}^{J} q(\omega_j,\theta,\phi) = \prod_{j=1}^{J}\frac{1}{\mathbf{v}^H(\omega_j,\theta,\phi)\mathbf{R}_{\mathbf{xx}}^{-1}(\omega_j)\mathbf{v}(\omega_j,\theta,\phi)} \tag{6.5}$$

where ω_j is the j^{th} narrowband component of the detected source signature.

This aggregated power spectrum is then searched over the incident angles θ and ϕ and the angles which maximize this function are determined to be the DOA angles of the source for that snapshot. This procedure is repeated every 1 second while the source is heard in order to produce successive DOA estimates which can be interpolated to form the trajectory of the source.

6.3 System Implementation Overview

The wideband Capon beamforming implementation consists of several steps as shown in Figure 6.1. In the first step, the data collected from all of the microphones in the array is passed to an FFT core every time a set of samples is available. Once enough samples have been passed to the FFT core (in our case 1024 samples) it calculates the FFT of the time series for each microphone. Next, these FFT coefficients are collected and a rank 1 spatial covariance matrix for each of the FFT bins of interest (frequencies in the frequency range of the detected source) is computed. These covariance matrices are then stored in the RAM of the EMS board. The next time these rank 1 covariance matrices are computed, they are instead added to the ones in RAM and the result is stored in the same location. This continues until K rank 1 covariance matrices have been averaged to form accurate estimates of the spatial covariance matrix for each frequency bin of interest. Once these full rank covariance matrices are obtained, they are inverted using a QR Factorization based matrix inversion algorithm [43] implemented in custom logic. Lastly, the narrowband Capon power spectra over all possible azimuth incidence angles (with a user-defined resolution) are formed for all of the narrowband frequencies associated with the classified source. They are then geometrically averaged over the frequencies of the classified source to create a single wideband power spectrum. This power spectrum is then searched over the angles for which it was computed to find the angle which corresponds to the maximum power. This angle is then taken as the DOA of the detected source. This procedure is done periodically at a rate of 1 estimate per second to produce a sequence of DOA estimates. These successive DOA estimates of the source are then used to form the source's trajectory relative to the microphone array.

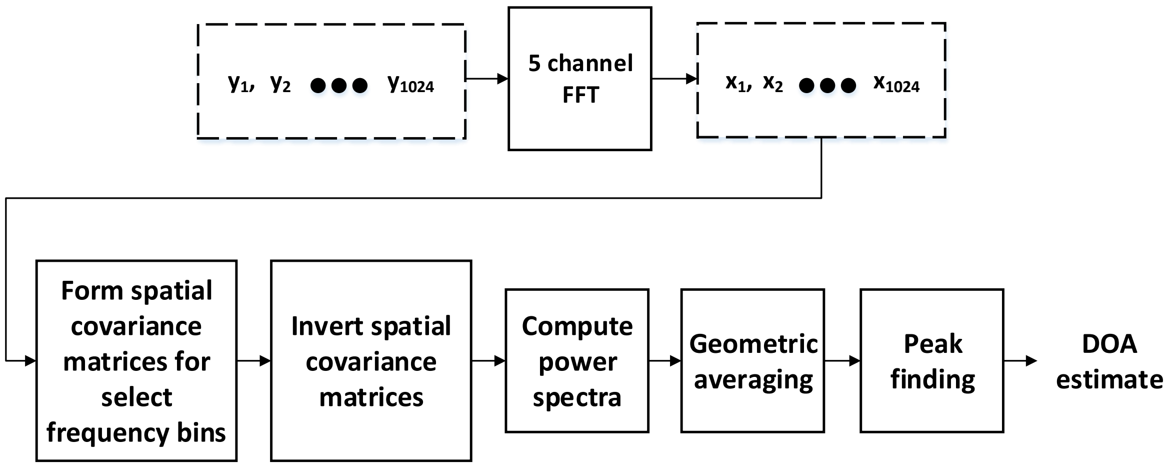


Figure 6.1: Wideband Capon Beamforming Data Flow

The specific steps in the process are described in the following subsections.

6.3.1 Data Acquisition

When a source is detected, the data from all of the InvenSense ICS-43432 microphones attached to the array is read simultaneously (using the same clock signal) at a rate of 48,000 samples per second. The 24-bit samples acquired by the microphones are clocked into the FPGA 1 bit at a time. Once a full set of samples (1 sample from each microphone) is obtained, they are concatenated together into a single logic vector and applied to a multi-channel FFT core which computes a 1024 point FFT on the data from each microphone. It then outputs the results of the FFT for each sequence one frequency bin a time to a first-in first-out (FIFO) type buffer starting with the lowest frequency and ending with the highest.

6.3.2 Calculating Spatial Covariance Matrices

Whenever a set of FFT outputs are available in the buffer, they are read by the spatial covariance matrix calculating state machine implemented in custom logic on the Artix 7 FPGA. Since the sets of values (one complex value for each microphone in the array) in the FIFO buffer are ordered by the frequency bin for which they were generated, this state machine keeps reading in sets of values from the buffer until it reads in a set that is associated with the first frequency of interest for the detected source. It then orders these values into a vector which is then multiplied by its transpose to form a rank 1 complex matrix. This matrix is then added element-wise to the matrix that is already stored in RAM for this frequency bin. Once a specific number K of rank 1 matrices for all frequency bins have been summed, the entries of all the matrices in RAM are normalized by K using a divider core (in our tests K was set to 32). The resulting full rank covariance matrices are output to another FIFO buffer 1 entry at a time. After these matrices are output, they are zeroed out in RAM in order to make room for the next set of covariance matrices that will be computed for the next DOA estimation segment.

6.3.3 Matrix Inversion

When data is available in the spatial covariance FIFO buffer, it is read in by the matrix inversion state machine (implemented as custom logic in the Artix 7 FPGA) until an entire covariance matrix is read in. It then converts each entry in the matrix into double precision using a Xilinx core in order to avoid possible overflow/underflow issues during the inversion process. It then computes the QR factorization of the matrix using a state machine and the modified Gram-Schmidt method outlined in [43].

The modified Gram-Schmidt method used to find the orthogonal $\mathbf{Q}$ matrix and the upper triangular $\mathbf{R}$ matrix from a full rank matrix $\mathbf{A} \in \mathbb{R}^n$ is implemented using a recursive algorithm [43]. In this algorithm, first the covariance matrix $\mathbf{A}$ is stored in a temporary matrix $\mathbf{T} = \mathbf{A}$. After that the algorithm iterates over the index i from 1 to n. At each iteration the algorithm first computes r_{ii}, the i^{th} diagonal entry of the $\mathbf{R}$ matrix, as the L^2 norm of $\mathbf{t}_i$, the i^{th} column of $\mathbf{T}$

$$r_{ii} = ||\mathbf{t}_i|| \tag{6.6}$$

Then, $\mathbf{q}_i$, the i^{th} column of the $\mathbf{Q}$ matrix, is computed by dividing the i^{th} column of the temporary $\mathbf{T}$ matrix by its L^2 norm

$$\mathbf{q}_i = \frac{\mathbf{t}_i}{||\mathbf{t}_i||} = \frac{\mathbf{t}_i}{r_{ii}} \tag{6.7}$$

After computing these two values for a value of i, the algorithm iterates over another index j from $i+1$ to n. During this iteration, the algorithm computes the off-diagonal entries of the i^{th} row in the $\mathbf{R}$ matrix using

$$r_{ij} = \mathbf{q}_i^H \mathbf{t}_j \tag{6.8}$$

where r_{ij} is the entry in the i^{th} row and j^{th} column of matrix $\mathbf{R}$. After each time an entry r_{ij} is computed, the algorithm updates the columns in $\mathbf{T}$ with their orthogonal projections unto subspace of $\mathbf{q}_i$

$$\mathbf{t}_j = \mathbf{t}_j - (\mathbf{q}_i^H \mathbf{t}_j \mathbf{q}_i) = \mathbf{t}_j - (r_{ij}\mathbf{q}_i) \tag{6.9}$$

Once all of the entries for the i^{th} column of $\mathbf{R}$ and all of the columns of $\mathbf{T}$ have been updated, the algorithm increments the first index i and goes back to calculating r_{ii} and $\mathbf{q}_i$. When it finishes iterating over all remaining values of i, the desired $\mathbf{Q}$ and $\mathbf{R}$ matrices are obtained.

After the decomposition, the upper triangular $\mathbf{R}$ matrix is inverted one column at a time using another state machine which executes a recursive inversion algorithm [43]. In this algorithm, the inverse of the upper triangular matrix $\mathbf{R}$ is first initialized to a zero matrix $\mathbf{R}^{-1} = 0$. Then, the algorithm iterates over the index j from 1 to n. At each iteration, the off-diagonal entries of the j^{th} column of $\mathbf{R}^{-1}$ are first recursively updated using

$$r_{ij}^{-1} = r_{ij}^{-1} + \sum_{k=1}^{j-1} r_{ik}^{-1} r_{kj} \quad \forall i \in [1 \ldots j-1] \tag{6.10}$$

where r_{ij}^{-1} is the entry in the i^{th} row and j^{th} column of $\mathbf{R}^{-1}$. Then, these entries are negated and normalized by the diagonal entry of $\mathbf{R}$

$$r_{ij}^{-1} = \frac{-r_{ij}^{-1}}{r_{jj}} \quad \forall i \in [1 \ldots j-1] \tag{6.11}$$

Finally, the diagonal entry of the j^{th} column of $\mathbf{R}^{-1}$ is computed by inverting the corresponding entry in $\mathbf{R}$

$$r_{jj}^{-1} = \frac{1}{r_{jj}} \tag{6.12}$$

Once the diagonal entry for the j^{th} column of $\mathbf{R}^{-1}$ is computed, the index j is incremented by 1 and the next iteration of the inversion algorithm starts by again updating the off-diagonal elements in $\mathbf{R}^{-1}$ according to (6.10). When the algorithm finishes iterating over all n columns of $\mathbf{R}$ using the index j, the actual inverse of the upper triangular matrix $\mathbf{R}$ is obtained.

After that, another state machine post multiplies the inverse of the upper triangular matrix with the Hermitian of the orthogonal $\mathbf{Q}$ matrix. This entire inversion process is repeated for each matrix in the spatial covariance buffer and the resulting inverse covariance matrices are stored in RAM in the same order in which they were processed (i.e. lowest frequency to highest). Since a full set of covariance matrices (one for each frequency of interest) is stored in the spatial covariance FIFO buffer every second, this state machine inverts all of these covariance matrices and stores them in RAM (overwriting the covariance matrices from the previous snapshot) every second.

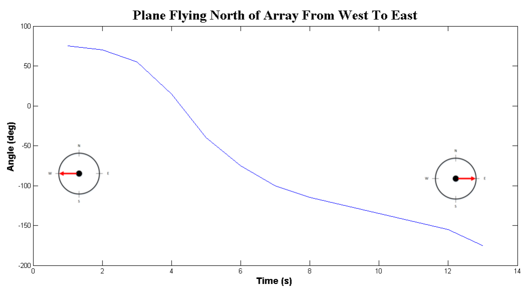
6.3.4 Beamforming

Once the inverses of the spatial covariance matrices are stored in RAM, a signal is sent to the Microblaze processor to initiate the next step of the beamforming. When this signal is received by the processor, it pre-multiplies each of the matrices by the Hermitian of its corresponding array steering vector for a single azimuth incidence angle and then post multiplies the result by the same steering vector thus producing a single complex value for each matrix. This is then repeated for each incidence angle over which the spectrum is to be searched resulting in an inverse of the narrowband Capon power spectrum for each frequency bin of interest. These inverted power spectra are then multiplied together to produce a single inverted wideband power spectrum. This inverted power spectrum is then searched for the minimum value (which is the reciprocal of the maximum value of the power spectrum) and the angle of arrival at which this minimum value occurs is declared as the angle of arrival of the source of interest. This procedure is repeated every time a DOA estimate is needed and the resulting sequence of DOA estimates is then logged so that it may later be used to generate the trajectory of sources of interest.

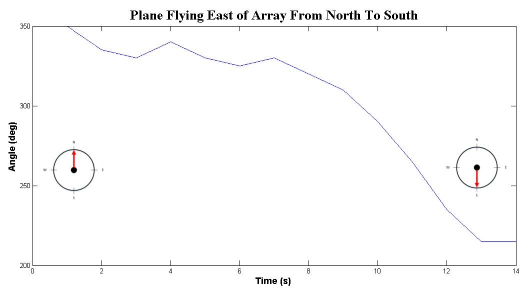
6.4 Field Test Results

A controlled full system test was also conducted using a simple 5 microphone wagon-wheel array as shown in Figure 1.1. The array was set up at the Christman Airfield on August 12, 2016 and an electric model airplane was flown in a straight line near the array on a pre-determined heading relative to the array with the start and end DOA angles (relative to the array) known in advance. The DOA estimates produced by the array during this test were then recorded and plotted to create an estimated trajectory of the plane which was compared with the heading at which the plane was flown. This was done several times with different headings relative to the array. During the test, only the azimuth DOA angle was estimated and the resolution of the estimates was 5 degrees. Unfortunately, the exact locations of the airplane could not be recorded during the tests due to technical issues with the GPS on the model plane. This prevented us from knowing the exact location of the plane during it's flight and thus a comparison of the actual DOA of the plane signal to the one estimated by the EMS could not be performed.

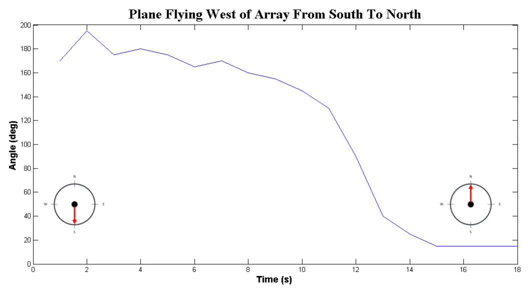
Figure 6.3 shows the tracks formed from the wideband Capon beamforming DOA estimates for several different headings at which the model plane was flown. The compasses on these figures show the actual starting and ending locations (relative to the array) of the model airplane for each track. These plots show that the DOA estimates were able to capture the general direction the model plane was flying in even though a few estimates were erroneous. These errors were likely caused by interference signals, such as wind, when the signal-to-noise ratio (SNR) was low due to the plane being far away from the array. Even with these errors, the trajectory of the source can be easily seen and this is sufficient for our application as we simply want to know in which direction the planes and helicopters that we detect are flying. This general heading along with a deployment map of the EMS systems should be enough in order to establish whether a classified source detected by one system is the same as a source detected by a neighboring system.



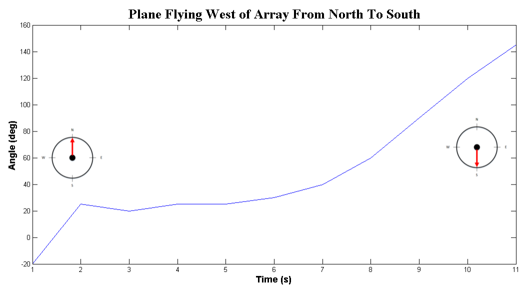
(a) North of array flying West to East



(b) East of array flying North to South



(c) West of array flying South to North



(d) West of array flying North to South

Figure 6.3: Azimuth DOA estimates of model plane on 4 different passes

6.5 Conclusion

In this chapter a wideband Capon beamforming method was reviewed which is used in the EMS system to form tracks for the sources classified by the SCST algorithm. This beamforming method is based on decomposing a signal captured by a microphone array into narrowband component and finding the linear filter for each component that would produce the minimum variance unbiased estimate of the narrowband signal. This is done for a number of steering vectors and then the power spectra of these components is computed for those vectors. These power spectra are then geometrically averaged and the DOA estimate is determined based on the location of the peak in the resulting power spectrum.

This algorithm was implemented mostly in custom logic due to the strict timing constraints on DOA estimation. The formation of the spatial covariance matrices and the calculation of their inverses needs to be done every estimation period and for every narrow-band frequency component of the source of interest. Since estimation periods need to be fairly small in order to have a decent track resolution for fast moving sources like planes and helicopters, these matrix operations needed to be done in a pipelined fashion in custom logic on the FPGA to ensure the DOA estimates are computed on time. The formation of the power spectra from these matrices (i.e. multiplication by steering vectors) and the subsequent geometric averaging of the spectra and maximization of the result are done in software running on a soft-core processor instantiated in the FPGA fabric. It is done this way because these calculations are simple enough that the soft-core processor can execute them nearly as fast as custom logic and since these steps do not lend themselves well to parallel execution there was not much performance to be gained by implementing them in hardware.

The field tests performed with the system show that it is able to track airborne acoustic sources. Although the accuracy of the DOA estimates produced by the system was not evaluated, the tests showed that the system is able to at least determine the general heading of an aircraft (i.e. flying to the West of the array from South to North) as it flies by the array which is sufficient for our purposes. Further tests will need to be conducted in order to evaluate the performance of the track formation algorithm on actual aircraft sources (e.g. helicopters, jets, etc.) and its performance in more realistic environments where interference sources are present.

CHAPTER 7
CONCLUSIONS AND FUTURE WORK

7.1 Conclusions and Discussion

The work in this thesis focused on the implementation of algorithms designed to detect, classify and track transient acoustic sources that can be commonly found in national parks across the U.S. More specifically, the SCST algorithm was implemented on the EMS system for detecting and classifying man-made acoustic sources like jets and helicopters. This algorithm was chosen due to its superior performance on this type of data. It was implemented on an Artix 7 FPGA and this implementation was benchmarked against the original MATLAB implementation of the algorithm. Furthermore, a full system test of the EMS system running this algorithm was conducted at Indian Pass in the Lake Mead National Recreation Area. The system was trained on signals captured over 48 hours (from July 5th to July 7th of 2016) at this location and then tested for 48 more hours (from July 18th to August 20th of 2016) at the same location. The tests revealed that the system operated as expected but only achieved a detection performance of 73.7% detection and 26.3% flase alarm at the knee point of the ROC curve. Also, the system frequently misclassified jet sources as helicopters and also classified most of the false detections as jets. This was likely due to the lack of robust training data available for the jet class and also due to the difficulty of separating strong wind from other source classes. Therefore, some parameters (e.g., sparse coding dictionary, signal and quiescent detection thresholds, etc.) will need to be fine tuned and more comprehensive training data needs to be collected in order to improve detection and classification performance.

The geometric averaging based wideband extension of the Capon beamformer was also implemented on the EMS in order to track acoustic sources that were detected and classified. In this implementation, the algorithm would be run on recorded data and produce a DOA estimate for the source every second. These DOA estimates would then be used to extrapolate a rough trajectory (in terms of angles relative to the microphone array) of the source. This algorithm was tested on August 12, 2016 at the Christman Airfield located west of Ft. Collins, Colorado. The test was conducted by flying a model plane at pre-determined trajectories near the microphone array and EMS system running this algorithm. The DOA estimates provided by the EMS were then used to create approximations of these trajectories. This test showed that the approximated trajectories were able to capture the general heading of the airplane but more precise tests will need to be conducted to evaluate the accuracy of the DOA estimates themselves.

7.2 Future Work

Though the initial testing results have been promising and the EMS implementation of SCST was shown to be comparable to the MATLAB implementation, there are a number of areas where the SCST algorithm itself can be improved.

1. **Different data pre-processing:** Currently, the EMS system uses 1/3 octave energies as inputs for the SCST algorithm. This is because historically NPS has used this data format for manually detecting and classifying acoustic sources. However, although this format is very space efficient, it is not the best for preserving discriminatory features between different classes. This is because there are only 33 octave bins for the full audible spectrum (from 12.5 Hz to 20 kHz) and only 11 of those bins are actually used to encode source signatures as our sources of interest range only from 100 Hz to 1 kHz. Additionally, as currently implemented, it finds the energy in each band over a one second period which could be too long for fast moving sources. It therefore makes sense to use a different filter bank to obtain more discriminatory features at finer time resolution. For example, a wavelet packet decomposition would result in an even breakdown of the spectrum into as many frequency bins as desired and then only the frequency bins in which the source signatures are known to lie can be used as inputs into the rest of the detection and classification algorithm.

2. **Improve Interference Rejection:** As mentioned in the Section 5.5.2, when there is strong wind which saturates the microphones, the log-likelihood ratios for signal detection tend to go up. This is because strong wind has a broadband signature that is difficult to differentiate from a combination of wind and a source signal. This in turn makes it difficult to tune the sparse coding dictionary so that only the effects of the wind are eliminated. Such a problem would also arise if any kind of interference that is not encountered during training or any kind of interference that has a significant overlap in the frequency domain with the sources of interest were present. One way to reduce this effect would be to have a pre-filter on the EMS that is calibrated to attenuate signals outside the frequency range of the sources that the system is trained to detect. However, this filter would still not completely get rid of the problem in the cases when the interference is present in the same frequency bins as the sources. In these cases, some type of thresholding can be used to prevent observations which have this type of interference present from increasing the CUSUM variables. In other words the $b_p(n)$ and $t_p(n)$ variables can be first compared to some user-defined threshold and only added to the cumulative B_p and T_p variables if they exceed the threshold. This way, even when some interference that passed the sparse coding phase looks more like a signal than pure noise, if it does not look enough like one of the signal classes (i.e. the log-likelihood for all the classes is below some threshold) then it will not increase the CUSUM variables. Like the other variables in SCST, this threshold would be tuned according to training data in order to filter out just the interference without significantly affecting the detection and classification of source events.

3. **Fuse Algorithms:** As mentioned earlier, the SCST and beamforming algorithm implementations were tested separately in order to minimize the impact of outside variables. However, when fully operational, the EMS system will need to run these algorithms in tandem and as such they will need to be integrated into a single code base. This will require some modifications of the beamforming implementation in order to

buffer data for a detected source until it is classified as the frequencies used by the beamforming algorithm will depend on the class of the source. Also, it will require the development of some sort of scheduling mechanism in order to ensure that the beamforming algorithm runs in parallel with the SCST algorithm but only when the SCST algorithm has detected a source. Since both algorithms are partially implemented on a Microblaze processor, this will require either 2 such processors and synchronization logic between them or some sort of scheduler to ensure that both algorithms execute within their respective timing constraints.

4. **Remote Data Collection:** Since the end goal of the EMS is to have autonomous monitoring of many different locations in various parks managed by NPS, it is essential that these systems are able to route the data they gather back to a central location for further analysis. This will involve developing networking and routing code for the EMS systems so that they will be able to reliably route reports to a central processing location.

5. **Remote Training:** In addition to the previous requirement of wireless transmission of data, it is also desirable that the systems support a wireless reprogramming capability so that the detection and classification parameters of any system can be remotely updated from a central location. With this capability, the systems will only need to be deployed once to a location and then can be trained and programmed remotely, thereby significantly cutting down deployment time and effort. Furthermore, with this capability, the detection, classification and tracking algorithms can be easily updated as needed without having to gather all the nodes that are already deployed to reprogram them and then deploy them again.

6. **Expert Evaluation and Benchmarking:** Although several validation tests have been run on the EMS board as described in this work, this system's performance has not been evaluated by any expert in noise monitoring. It will thus be valuable

to benchmark the performance of this system for detecting, classifying and tracking transient acoustic signals against other similar systems and also against the detection and classification performance which can be achieved by a trained human technician on similar data. These benchmarking tests would allow for a fair evaluation of the performance of the EMS board when compared to existing methods.

7. **Comprehensive Field Tests:** So far the EMS has only been tested in one location and on only 3 signal types (2 source types and 1 interference). While the results of this test were promising, this system was designed to work in many different acoustic environments with a wide variety of source and interference signals. Therefore, additional testing of this system will need to be performed at different locations in a number of national parks in order to better ascertain how effective this system will be at noise monitoring.

REFERENCES

[1] B.-H. Juang and L. Rabiner, “Mixture autoregressive hidden Markov models for speech signals,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 6, pp. 1404–1413, Dec 1985.

[2] A. A. M. Abushariah, T. S. Gunawan, O. O. Khalifa, and M. A. M. Abushariah, “English digits speech recognition system based on hidden Markov models,” pp. 1–5, May 2010.

[3] J. G. Wilpon, L. R. Rabiner, and T. Martin, “An improved word-detection algorithm for telephone-quality speech incorporating both syntactic and semantic constraints,” *AT T Bell Laboratories Technical Journal*, vol. 63, no. 3, pp. 479–498, March 1984.

[4] A. Digulescu, M. Paun, C. Vasile, T. Petrut, D. Deacu, C. Ioana, and R. Tamas, “Electrical arc surveillance and localization system based on advanced signal processing techniques,” pp. 426–430, May 2014.

[5] L. P. S. Fernandez, A. R. Ruiz, and O. B. Pogrebnyak, “Noise monitoring of aircrafts taking off based on neural model,” pp. 1–8, Sept 2009.

[6] C. Han, P. Willett, B. Chen, and D. Abraham, “A detection optimal min-max test for transient signals,” *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 866–869, Mar 1998.

[7] V. Bruni, S. Marconi, and D. Vitulano, “Time-scale atoms chains for transients detection in audio signals,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 420–433, March 2010.

[8] B. Thoshkahna, F. X. Nsabimana, and R. R. Kalpathi, “A transient detection algorithm for audio using iterative analysis of STFT,” pp. 203–208, 2011.

[9] S. W. Foo and T. Yap, "HMM speech recognition with reduced training," vol. 2, pp. 1016–1019, Sep 1997.

[10] T. S. Brandes, "Feature vector selection and use with hidden Markov models to identify frequency-modulated bioacoustic signals amidst noise," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 6, pp. 1173–1180, Aug 2008.

[11] A. Kundu, G. C. Chen, and C. E. Persons, "Transient sonar signal classification using hidden Markov models and neural nets," *IEEE Journal of Oceanic Engineering*, vol. 19, no. 1, pp. 87–99, Jan 1994.

[12] B. Chen and P. Willett, "Superimposed HMM transient detection via target tracking ideas," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 37, no. 3, pp. 946–956, Jul 2001.

[13] B. Chen and P. Willett, "Detection of hidden Markov model transient signals," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 4, pp. 1253–1268, Oct 2000.

[14] L. Angrisani, P. Daponte, and M. D'Apuzzo, "Wavelet network-based detection and classification of transients," *IEEE Transactions on Instrumentation and Measurement*, vol. 50, no. 5, pp. 1425–1435, Oct 2001.

[15] N. Wachowski and M. R. Azimi-Sadjadi, "Characterization of multiple transient acoustical sources from time-transform representations," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 9, pp. 1966–1978, Sept 2013.

[16] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using Gaussian mixture speaker models," *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 1, pp. 72–83, Jan 1995.

[17] P. Somervuo, A. Harma, and S. Fagerlund, "Parametric representations of bird sounds for automatic species recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 6, pp. 2252–2263, Nov 2006.

[18] S. Chu, S. Narayanan, and C. C. J. Kuo, "Environmental sound recognition with time-frequency audio features," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 6, pp. 1142–1158, Aug 2009.

[19] N. Wachowski and M. Azimi-Sadjadi, "Detection and classification of nonstationary transient signals using sparse approximations and bayesian networks," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 22, no. 12, pp. 1750–1764, Dec 2014.

[20] S. Kay, *Fundamentals of Statistical Signal Processing, Volume 2: Detection Theory.* Prentice Hall PTR, 1998.

[21] I. Ben-Gal, "Bayesian networks," *Encyclopedia of statistics in quality and reliability*, 2007.

[22] N. Wachowski, "Characterization of multiple time-varying transient sources from multivariate data sequences," Ph.D. dissertation, Colorado State University. Libraries, 2007.

[23] T. Pham and B. M. Sadler, "Adaptive wideband aeroacoustic array processing," pp. 295–298, Jun 1996.

[24] H. Wang and M. Kaveh, "Coherent signal-subspace processing for the detection and estimation of angles of arrival of multiple wide-band sources," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 4, pp. 823–831, Aug 1985.

[25] J. A. Cadzow, "Multiple source location-the signal subspace approach," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 7, pp. 1110–1125, Jul 1990.

[26] M. Azimi-Sadjadi, A. Pezeshki, and N. Roseveare, “Wideband DOA estimation algorithms for multiple moving sources using unattended acoustic sensors,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, no. 4, pp. 1585–1599, Oct 2008.

[27] R. Schmidt, “Multiple emitter location and signal parameter estimation,” *IEEE Transactions on Antennas and Propagation*, vol. 34, no. 3, pp. 276–280, Mar 1986.

[28] M. R. Azimi-Sadjadi, A. Pezeshki, L. L. Scharf, and M. E. Hohil, “Wideband DOA estimation algorithms for multiple target detection and tracking using unattended acoustic sensors,” *Proc. SPIE*, vol. 5417, pp. 1–11, 2004.

[29] *Model 831 Sound Level Meter Technical Reference Manual*, Larson Davis, 2006.

[30] “Electroacoustics - octave-band and fractional-octave-band filters - part 1: Specifications,” *IEC 61260-1:2014*, 2014.

[31] “Specification for octave, half-octave, and third octave band filter sets,” *ANSI S1.11-2004*, 2004.

[32] *Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics*, Xilinx, 2015.

[33] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, “The flooding time synchronization protocol,” pp. 39–49, 2004.

[34] *ICS-43432 Datasheet*, InvenSense, 2016.

[35] M. Aharon, M. Elad, and A. Bruckstein, “K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[36] P. R. Gill, A. Wang, and A. Molnar, “The in-crowd algorithm for fast basis pursuit denoising,” *IEEE Transactions on Signal Processing*, vol. 59, no. 10, pp. 4595–4605, Oct 2011.

[37] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," vol. 1, pp. 40–44, Nov 1993.

[38] L. Scharf, *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis.* Addison-Wesley, 1991.

[39] M. R. Azimi-Sadjadi, J. Kopacz, and N. Klausner, "K-SVD dictionary learning using a fast OMP with applications," pp. 1599–1603, Oct 2014.

[40] J. N. Tsitsiklis, "Extremal properties of likelihood-ratio quantizers," *IEEE Transactions on Communications*, vol. 41, no. 4, pp. 550–558, Apr 1993.

[41] "Preferred numbers – series of preferred numbers," *ISO 3:1973*, 1973.

[42] H. Van Trees, *Detection, Estimation, and Modulation Theory, Optimum Array Processing.* Wiley-Interscience, 2002.

[43] A. Irturk, B. Benson, and R. Kastner, "An efficient FPGA implementation of scalable matrix inversion core using QR decomposition," *UCSD Technical Report, CS2009-0938*, 2009.