12-1-2016

Implementation and Performance Evaluation of Acoustic Denoising Algorithms for UAV

Ahmed Sony Kamal Chowdhury
University of Nevada, Las Vegas, sonyc@unlv.nevada.edu

IMPLEMENTATION AND PERFORMANCE EVALUATION OF ACOUSTIC DENOISING ALGORITHMS FOR UAV

By

Ahmed Sony Kamal Chowdhury

Bachelor of Science - Electrical and Electronics Engineering
North South University, Dhaka, Bangladesh
2013

A Thesis Submitted in Partial Fulfillment
of the Requirements for the

Master of Science in Engineering– Electrical Engineering

Department of Electrical and Computer Engineering
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
December 2016

UNLV | GRADUATE COLLEGE

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

August 10, 2016

This thesis prepared by

Ahmed Sony Kamal Chowdhury

entitled

Implementation and Performance Evaluation of Acoustic Denoising Algorithms for UAV

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Engineering - Electrical Engineering
Department of Electrical and Computer Engineering

Venkatesan Muthukumar, Ph.D.
Examination Committee Chair

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Interim Dean

Emma Regentova, Ph.D.
Examination Committee Member

Sahjendra Singh, Ph.D.
Examination Committee Member

Ajoy Datta, Ph.D.
Graduate College Faculty Representative

ABSTRACT

Unmanned Aerial Vehicles (UAVs) have become popular alternative for wildlife monitoring and border surveillance applications. Elimination of the UAV's background noise and classifying the target audio signal effectively are still a major challenge. The main goal of this thesis is to remove UAV's background noise by means of acoustic denoising techniques. Existing denoising algorithms, such as Adaptive Least Mean Square (LMS), Wavelet Denoising, Time-Frequency Block Thresholding, and Wiener Filter, were implemented and their performance evaluated. The denoising algorithms were evaluated for average Signal to Noise Ratio (SNR), Segmental SNR (SSNR), Log Likelihood Ratio (LLR), and Log Spectral Distance (LSD) metrics. To evaluate the effectiveness of the denoising algorithms on classification of target audio, we implemented Support Vector Machine (SVM) and Naive Bayes classification algorithms. Simulation results demonstrate that LMS and Discrete Wavelet Transform (DWT) denoising algorithm offered superior performance than other algorithms. Finally, we implemented the LMS and DWT algorithms on a DSP board for hardware evaluation. Experimental results showed that LMS algorithm's performance is robust compared to DWT for various noise types to classify target audio signals.

ACKNOWLEDGMENTS

I would like to thank Dr. Venkatesan Muthukumar for his excellent guidance throughout my thesis completion. He provided great support to me. He has helped me by answering my questions and coming up with valuable suggestions for improvement of the performance in my project.

I would also like to thank my thesis committee members, Dr. Sahjendra Singh, Dr. Emma Regentova and Dr. Ajay. K. Dutta, for providing valuable guidance and reading this thesis report.

I am grateful for the support from my family and friends; they always give me encouragement and confidence.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER I: INTRODUCTION

1.1 Overview

Recently various Unmanned Aerial Vehicles (UAVs) application in urban and non-urban scenarios are being employed. One such application is in remote monitoring of wildlife and humans. Commonly, vision-based sensors and techniques are used for wildlife, human monitoring. However, these techniques require a clear line of sight, uniform illumination, and occlusion-free environments for effective operation. The use of acoustic techniques overcomes these problems that vision-based sensors encounter. However, acoustic techniques have different constraints such as noisy environments, cluttering of audio signals, etc. Audio signals are usually corrupted by environment noise, equipment noise, wind noise etc. Audio denoising techniques aim at minimizing these noise while retaining the required signals. The basic idea behind this thesis is to explore denoising algorithm to extraction target audio signals from the noisy signals. There are various are various denoising algorithm to help restore/extract an audio signals from noisy signals. Selecting the appropriate algorithm plays a major role in detection and classification of the target signal. The denoising algorithm tend to be specific. For example, a algorithm that is used to denoise vehicle audio signals may not be suitable or providing optimum performances for denoising of bird or any other class of audio signals. In this thesis, a study of various denoising algorithms and their performance are evaluated for various classes. In order to evaluate the performance of various denoising algorithms, an audio dataset with four classes: animals, humans, birds and vehicles and a noise dataset contains Additive White Gaussian Noise (AWGN), UAV Propeller and Motor (PM) noises and DJI drone noises were developed.

Major audio signal denoising methods are based on reduction in time-frequency signal domain. It has been suggested by Marmaroli and Falourd [1], due to low Signal to Noise Ratio (SNR) caused by the wind, vibration and UAV's own propeller noise when UAV operates, the acoustic denoising process encounters a number of substantial challenges in order to detect and classify target class.

In this research, we implement and evaluate four popular denoising algorithms; time-frequency Block-Thresholding (BTH), Wiener Filter (WF), Least Mean Square (LMS) and Discrete Wavelet transform (DWT) denoising algorithm due to their robust performance and low complexity of implementation [2-4]. These denoising algorithms are evaluated based on the following performance criteria: 1) Average Signal to Noise Ratio (SNR) value, 2) Segmental SNR, 3) Log Likelihood Ratio (LLR) and 4) Log Spectral Distance (LSD) [4]. The noise sources considered are Additive White Gaussian Noise (AWGN) as well as actual recordings of a UAV propeller noise and on-board motor noise.

Initial simulations shows that the use of adaptive noise cancelling algorithm LMS and DWT offer better performance hence were implemented in hardware Texas Instrument's TMS320C5535 DSP board implementation is feasible [5]. The use of adaptive denoising algorithm will reduce the noise of the target audio and will improve classification accuracy for target class of UAV.

MATLAB simulation of various denoising algorithms show performance in SNR, SegSNR, LSD and LLR with respect to various noise sources and noise levels.

1.2 Motivation

The motivation behind this thesis is the growth in adoption of UAV in wildlife monitoring, border patrol and residential surveillance application, in order to recognize and classify the target object accurately based on their acoustic signature. However, due to interference of surrounded acoustic sources, UAV faces substantial challenges during detection, classification and localization. Moreover, passive denoising techniques like as barriers, enclosure and silencers to reduce the surrounding unwanted noise are expensive. This led us to focus on active denoising by employing some active denosing methods.

1.3 Objective

This research focuses on several aspects of acoustic denoising and its applications. The major objective of this thesis is as follows:

- Theory and study of the existing acoustic denoising algorithms.
- Adopt existing denoising algorithms to improve SNR and quality of the target signal.
- Implement and evaluate denoising algorithm in MATLAB software and hardware implementation on Texas Instrument's TMS320C5535 DSP board.
- Finally, evaluating denoising algorithm performance on acoustic classification application, before and after denoising.

1.4 Organization of Thesis

The thesis is organized as follows:

Chapter 2 discuss background and literature review of the existing denoising algorithms. A brief explanation of audio signal fundamentals, audio features extraction and classification methods for audio signals is introduced.

Chapter 3 discuss the theory of selected audio denoising algorithms and metrics used for performance evaluation.

Chapter 4 explains the software implementation details of denoising algorithms is MATLAB, Texas Instrument's TMS320C5535 DSP environment and implementation for classification respectively.

Chapter 5 presents the software simulation and hardware implementation result.

The thesis concludes with chapter 6 which summarizes the results, conclusions and discussions for future work.

CHAPTER II: BACKGROUND AND LITERATURE REVIEW

2.1 Audio Signal

In this section, an overview of audio signal basics and time-frequency approach to represent acoustic signals is discussed. Audio signals consist of complex waveforms. Signals are combinations of different fundamental frequencies with unique harmonic content created by particular instruments or other audio sources. It is nearly impossible to extract the harmonic content of an audio signal from the time-domain waveform. However, taking the Fourier Transform (FT) of the signal, and looking at the spectrum reveals the frequency components that make up the signal. Frequency spectrum is the basis for distinguishing the desired signal from the noise [3]. In many cases, most harmonic information is hidden in the frequency content of the signal. Basically, the frequency spectrum of a signal is the frequency components, also called the spectral components of the signal. Frequency spectrum reveals what frequencies exist in the signal. In the context of audio processing, signals are irregular and non-stationary, but temporally stationary over a short period of time. However, the FT gives no information regarding where in time those spectral components appear. Therefore, the FT is not a suitable technique for a non-stationary signal. To circumvent this lack of locality property, the Short-Time Fourier transform (STFT) is applied to represent the signal both in time and frequency domain [3-4]. STFT is a revised version of Fourier transform where the signal is divided into small segments (portions) that are small enough to be assumed as stationary segments of the signal. STFT is defined as:

$$STFT(t, f) = \int_{-\alpha}^{\alpha} f(t) w(t-\tau) e^{-j2\pi ft} dt \qquad (1)$$

The only difference between FT and STFT is that the audio signal *f(t)* in the STFT is multiplied with a sliding window function *w(t)*. The window function, *w(t)* to set chunks of the stationary signal. To analyze non-stationary signals when there is a need for information of the time localization, a spectrogram is widely used. The spectrogram is a joint time-frequency two-dimensional plot. The spectrogram uses the STFT to take Fourier transform of a small enough time segments of the signal [3]. The spectrogram is defined as:

$$P_{sp}(t, f) = |STFT(t, f)|^2 \quad (2)$$

The length of the time segments is chosen so that the signal hopefully is stationary within each frame. The spectrum for each time segment is the absolute value squared to get the energy distribution of the signal.

2.2 Noise

Noise can be defined as any unwanted component inside a signal, either random or deterministic. Noise interferes with the optimal reproduction of desired signal in a system. These unwanted signals arises from a variety of sources. Some examples of noise are a UAV's propeller, engine motor noise, or environmental noise such as wind. A certain degree of noise is always present in any electronic device that transmits or receives a signal. Furthermore, White noise is a random signal with a flat power spectral density for which the frequency and power spectrum are constant and independent of frequency. The name "white noise" comes from the similarity to the white light which has equal quantities of all colors. Gaussian noise is statistical in nature. Its probability density function is equal to that of a normal distribution, which is otherwise called a Gaussian distribution. A special case of Gaussian noise is white Gaussian noise, in which the

values are always statistically independent. In this research, the Additive White Gaussian Noise (AWGN), UAV's propeller, and engine noise when operating are considered [1-5].

2.3 Audio Denoising Algorithms

Audio denoising aims at removing noise while retaining the underlying original signals. Some applications of audio denoising include music and speech restoration. Audio denoising techniques are basically divided into two groups: Diagonal estimation techniques and non-diagonal estimation techniques. To attenuate the noise from audio signals diagonal time-frequency audio denoising algorithms process each spectrogram coefficient independently. The drawback of these algorithms is they have a limited performance, denoised signal contain noise, denoised sound is contaminated and the audio perception is degraded due to the superposition of noise. To overcome these drawbacks non-diagonal estimation techniques are required. There are various types denoising algorithm exist such as Spectral Subtraction, LMS, Recursive Least Square (RLS), DWT, BTH, WF etc. However, in this work we select to LMS, DWT, BTH and WF due to ease of implementation and robust performance [2-4].

Least Mean Square (LMS) is an adaptive iterative algorithm. There are numerous adaptive algorithms available for audio denoising applications such as Recursive Least Square (RLS), LMS, Normalized Least Mean Square (NLMS), etc. Several authors [6-8] propose many variants of adaptive denoising algorithms. Furthermore, a number of adaptive algorithms have been proposed to improve the convergence rate, tracking speed, and weight adjustment. LMS adaptive algorithm has been chosen in this work due to its algorithmic simplicity, better stability,

low computation time, swift convergence rate and adaptive weight adjustment [9-11]. Figure. 2.3.1. Illustrates the system block diagram of the LMS adaptive denoising.

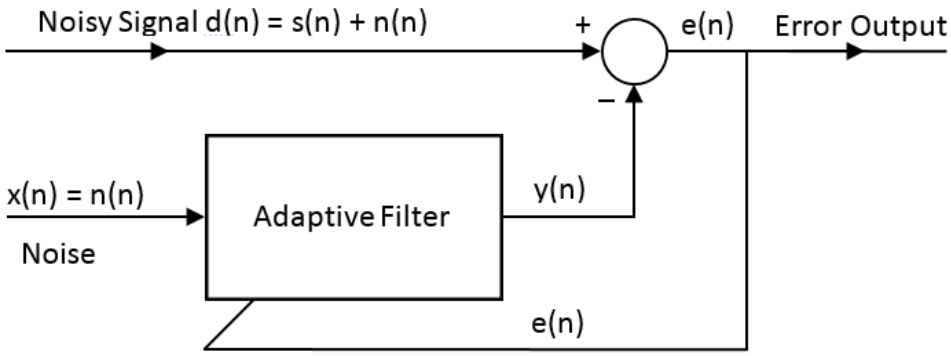


Figure 2.3.1 Block diagram of LMS adaptive denoising algorithm

Technically, an adaptive filter is a system that uses previously captured filter parameters to adjust and update the new parameter values for adapting to unknown statistical properties of signals and noises. The goal is to achieve optimal filtering over time. In the case of processing observed signals with unknown statistical properties, an adaptive filter can achieve satisfying results that are far better than other fixed parameter filters designed with universal methods. Adaptive filters are extensively applied in the field of communication to removing background noise in radar, sonar, control engineering, and biomedical science [12-13].

In time-frequency domain analysis of an audio signal, the windowed short-time Fourier transform (STFT) provides temporal information about the frequency content of signal. A drawback of the STFT is its fixed time resolution due to a fixed window length. The Wavelet Transform (WT), with its flexible time-frequency window, is an appropriate tool for the analysis of nonstationary signals like human speech. Wavelets have been utilized in a large number of fields including: acoustic, speech and music processing, image processing, telecommunications,

seismology, medicine, and biology. WT decomposes signals over translated and dilated mother wavelets. A mother wavelet is a function of time with finite energy and fast decay. The mathematical and detailed explanation of wavelet transform, optimum wavelet functions selection, wavelet decomposition level, proper threshold selection and reconstruction algorithm can be found in [14-18]. The CWT is computed by changing the scale of the mother wavelet, shifting the scaled wavelet in time, multiplying by the signal, and integrating over all times. The difference between the mother wavelet functions (e.g., Haar, Daubechies, Symlets, Coiflets, Biorthogonal, etc.) depend on how these scaling signals and the wavelets are defined. The representation of the signal is often redundant and that's the major drawback of CWT. To overcome this situation, researchers have proposed and discovered Discrete Wavelet Transform (DWT). Due to the orthonormal properties there is no information redundancy in DWT. In DWT decomposition, two wavelet decomposition (Analysis) filters, e.g., High Pass Filter and Low Pass Filter, are created, followed by down sampling by two, producing half of input data point of High and Low frequency [19-20]. The detailed coefficients represent the high frequency coefficients, and approximation coefficients represent the low frequency coefficients. On the other side, thresholding can be done in two different ways. The soft and hard thresholding methods are the most popular, and widely used to estimate wavelet coefficients in wavelet threshold denoising [21]. Figure. 2.3.2, presents the hard and soft thresholding signals.

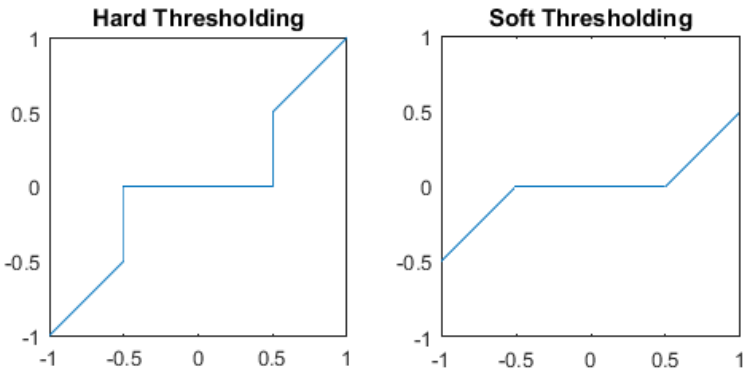


Figure 2.3.2 From the left, Hard and soft thresholding signals

For DWT hard thresholding sets all wavelet coefficients below the given threshold value equal to zero and exhibits artifacts. Soft thresholding smoothen the signal by reducing the wavelet coefficients by a quantity equal to the threshold value and modifies the signal energy [15-19]. The wavelet function and decomposition level also play an important role in the quality of the denoised signal.

The Wiener Filter (WF) was introduced by Norbert Wiener in the 1940s. A major feature was the use of a statistical model to estimate the signal. The WF algorithm developed by Lim and Oppenheim can reduce noise based on minimizing the Mean Squared Error (MSE) criterion in the frequency domain. WF is one of the most widely used tools in signal processing, especially for applications in signal denoising and source separation [22]. There are many improved algorithms based on Wiener filtering. However, WF algorithm has some practical deficiency

since the power spectrum of clean audio signal and additive noise cannot be obtained directly. In the Iterative Wiener Filter (IWF) algorithm, the power spectrum of the signal is estimated through the Linear Predictive Coding (LPC) analysis of noisy signals. IWF estimates the power spectrum of noise by processing the clean signal segments. The estimated noise power spectrum can thus be used to design the WF. Then LPC analysis is applied to the enhanced target signal. The steps above are repeated in iterations to obtain a more accurate speech power spectrum and design the optimal WF. However, there is one significant drawback of the IWF algorithm which is the lack of proper convergence criteria [23-24]. This will induce serious distortion to the estimated clean signal. The performance of these filtering techniques depends on the distortion level of the processed audio signal.

The Time-frequency Block Thresholding (BTH) algorithm was initially introduced by Cai and Silverman to statistically improve the asymptotic decay of diagonal thresholding estimators [25-27]. Depending upon the SNR considered, the audio signal denoising techniques are basically divided into two categories [26-27]:

- Diagonal Estimation Technique
- Non-Diagonal Estimation Technique

Diagonal time-frequency audio denoising algorithms reduce the unwanted noise by treating each window Fourier coefficient independently, with empirical power subtraction or thresholding operators. The drawback of this Diagonal Estimation Technique is that the denoised signal contains musical noise, the denoised signal is contaminated, and the audio perception is degraded due to the superposition of musical noise. To overcome these drawbacks, non diagonal estimation techniques are required [27]. In this work, non-diagonal block

thresholding audio denoising procedure is implemented. This block thresholding algorithm processes the STFT coefficient of a noisy signal. This algorithm reduces the Short Time Fourier Transform (STFT) coefficient of noisy signals by blocks through an identical attenuation factor over each block by using Stein Unbiased Risk Estimator (SURE) theorem [28-29]. The time-frequency audio denoising algorithms perform a parameterized filtering of spectrogram coefficients with empirically fixed parameters. In addition, block size and thresholding level in time-frequency signal representations are studied.

Next we discusses two concepts in acoustic application: feature extraction and classification. The common approach of audio classification is to extract discriminatory features from the audio data and feed them into a pattern classifier [30-31]. The better and more effective features that are extracted from audio signals, the higher the performance that will be achieved in the audio classification procedure. Therefore, feature extraction is the most important phase of the classification process. In feature extraction, some transformations are used to extract/select features that best represent the characteristics of the source of audio signal. A set of these extracted features is called a feature vector. Feature vectors may be represented in time, frequency, or time-frequency domain. The time-domain features are simply extracted by sampling the audio signal. Analyzing audio signals with time-domain features can be simple, although it is usually necessary to also analyze complex features in frequency-domain. Frequency based feature extraction methods include Fast Fourier Transform (FFT) and Mel-frequency cepstral coefficients (MFCC). MFCCs have been widely used in speech recognition, musical genre classification, speaker clustering, and in many other audio analysis applications. In this work, we extract MFCC based audio features [32]. Then classifiers allow division of the

feature space into regions, where each region corresponds to a certain class. The effectiveness of the overall classification procedure depends strongly on the choice of the optimal classifier with the ability to adapt to various classification problems. We use Support Vector Machine (SVM) and Naïve Bayes (NB) based classifier in this work, due to its generalization ability, and superior performance in various pattern classification tasks. SVM has been extensively used in many applications of pattern recognition such as image analysis, text and audio classification, speech recognition and speaker identification [33-34]. SVM uses a nonlinear kernel function to map a sample of classes. SVM maps classes into a high dimensional feature space. SVM learns to find the optimal separating hyper-plane, thus maximizing the margin of classes [35]. Compared to other classifiers that separate the data in its original space, such as DT (decision tree), Neural Network (NN), and Naive Bayes, SVM maps non-linear separable data into higher dimensional space, and performs separation in that space. To compare SVM classifiers performance, we implement Naïve Bayes (NB) classification algorithm [36-38]. NB classifier is one such framework that has been widely used in text and image classification. NB classification technique based on divide and conquer strategy. It assumes that each feature vector in the feature set is independently generated with identical distribution. Class labels are then predicted by maximizing the likelihood function for the posterior probability [39-41].

The effectiveness of the above mentioned MFCC features along with SVM and Naive Bayes based audio classification techniques are discussed in chapters III and implemented in IV.

In our evaluation of the above implemented denoising algorithms performance, four metrics were used. The first two are non-perceptual group in time-domain measures are respectively the SNR and Segmental SNR (SSNR) is defined by equation (3) and (4) respectively:

$$SNR = 10\log_{10}\frac{\sum x^2(i)}{\sum_{i=1}^{N}(x(i)-y(i))^2} \tag{3}$$

$$SegSNR = \frac{10}{M}\sum_{m=0}^{M-1}\log_{10}\sum_{i=Nm}^{Nm+N-1}\left(\frac{x^2(i)}{x(i)-y(i)^2}\right) \tag{4}$$

Where *x(i)* is defined as original clean audio signal, *y(i)* is noisy audio signal, *M* represents the number of frames and *N* denoted as number of samples per frame in the audio signal. The SNR is very sensitive to the time alignment of the original and distorted audio signal.

The rest two are perceptual group in frequency domain measures such as Log Likelihood Ratio (LLR) and Log Spectral Distance (LSD). The LLR considers an all-pole linear predictive coding (LPC) model of audio segments.

$$x[n] = \sum_{m=1}^{p} a(m)x[n-m] + G_x u[n] \tag{5}$$

The LLR is defined as:

$$LLR = \log\left(\frac{a_x^T R_x a_x}{a_y^T R_y a_y}\right) \tag{6}$$

Where a_x is the LPC coefficient of clean audio signal of *X[n]*, a_y is the corresponding vector of corrupted noisy audio signal with corresponding covariance matrices R_x and R_y.

The LSPD is referred to as log-spectral distortion, is a distance measure between two spectra, expressed in dB. Hence the LSPD is defined as:

$$LSPD = \sqrt{\frac{1}{2\pi}\int_{-\pi}^{\pi}\left[10\log_{10}\frac{P(\omega)}{\hat{P}(\omega)}\right]^2 d\omega} \qquad (7)$$

Where $P(\omega)$ *and* $\hat{P}(\omega)$ are power spectra of audio signal.

CHAPTER III: DENOISING ALGORITHMS

3.1 Least Mean Square

LMS adaptive filter mainly consist of two parts: (1) filtering process generating the output signal and error estimation error, and (2) adaptive process responsible for the automatic adjustment of filter tap weights [6-7]. If the reference signal is denoted as *X(n)*, filter output is *Y(n)*, error signal represents as *e(n)* and *d(n)* refers as expected signal. Consequently transversal filter order is *M* and the reference signal is *X(n)*, at time *n* then the filter output can be written as:

$$y(n)=\sum_{m=1}^{M} w_m(n)x(n-m+1) \tag{8}$$

Where W_m *(n,m=1,2,..M)* refers to each weighting coefficient for the input. Define *w(n)* and *x(n)* respectively as the weighting coefficient vector and reference signal vector of the adaptive filter. Equation (8) can be expressed in vector form [8-9]:

$$y(n)=x^T(n)w(n)=w^T(n)x(n) \tag{9}$$

Consequently the error signal can be expressed as:

$$e(n)=d(n)-y(n)=d(n)-w^T(n)x(n) \tag{10}$$

Then the weight updating recursive formula can be expressed as:

$$w(n+1)=w(n)+2\mu e(n)x(n) \tag{11}$$

Where μ is defined as step factor which is also known as convergence factor. This factor determines the rate of convergence of the filter. An initial filter output is calculated from the input response and an initial weighting coefficient. The error signal, which is the difference between the output and desired signal, is estimated. Then the weighting coefficient vector is updated using previous weighting coefficient, step factor, error and input response. The goal is to achieve better filtering over time which the noise statistically reduces to minimum level after some time. Thus equation (9), (10) and (11) represents as a full mathematical expression of LMS adaptive filter denoising algorithm.

3.2 Discrete Wavelet Transform

Wavelet denoising algorithm is popular method for audio denoising. The algorithm is different from parametric procedures in which all parameters must be estimated for a specific model which must be assumed a priori [14]. Discrete Wavelet Transform (DWT) is chosen in this research due to signal structure variations such as long harmonics and short transient [15]. A DWT of the noisy signal is computed by the time-frequency audio denoising procedures and processes the resulting coefficient to attenuate noise [18]. Consider the observe data:

$$X(t) = S(t) \oplus N(t) \qquad (12)$$

Where additive white noise is represents as *N(t)* with clean audio signal is *S(t)* as a function of time *t*. Let $W^{-1}(.)$ and $W(.)$ indicate the inverse and forward wavelet transform operator. Next assume $D(.\lambda)$, which indicates the denoising operators with soft thresholds λ. Then the goal of this algorithm is to wavelet denoise *X(t)* in order to recover $\hat{S}(t)$ as an estimate of *S(t)*. The following steps summarize the complete procedure:

$$Y = W(x) \tag{13}$$

$$Z = D(Y, \lambda) \tag{14}$$

$$\hat{S} = W^{-1}(Z) \tag{15}$$

Then the continuous–time wavelet transform *f(t)* can be written as:

$$CWT_{\psi} f(a,b) = W_f(b,a) = \left| a^{-\frac{1}{2}} \right| \int_{-\infty}^{\infty} f(t) \psi * \left(\frac{t-b}{a} \right) dt \tag{16}$$

Where, the dilating and translating coefficients are denoted as a, b∈ R, a≠ 0 individually. In order to receive the transformed signal with same energy at every scale, $|a^{-1/2}|$ is multiplied for energy normalization purpose. The mother wavelet function $\psi(t)$ is considered as a analysis function with a requirement that this function has a zero net area which implies that the transformation kernel of the wavelet transform is a absolutely support function [18-19]. Since *a* and *b* are continuous over *R*, the representation of signal is often redundant and this is a major drawback of CWT. However the original signal can be reassembled by sample version of $W_f(b,a)$. However by sample version of $W_f(b,a)$ the original signal can be reassembled. In dyadic grid, the signal is typically sample as $W_f(b,a)$, i.e., $b=n2^{-m}$, $m,n \in Z$. Then substituting the last one into equation (16), we obtained:

$$DWT_{\psi} f(a,b) = \int_{-\infty}^{\infty} f(t) \psi * (t) dt \tag{17}$$

Where $\psi_{m,n}(t)=2^{-m}\ \psi(2^m t-n)$ is dilated and translated version of mother wavelet $\psi(t)$. In this research, Daubechies mother wavelet function with soft thresholding is used [18]. By decreasing

the wavelet coefficient with an amount equal to the threshold value and adapting the signal energy, soft thresholding can be smoothen the signal.

3.3 Wiener Filter

The Wiener Filter (WF) is a well-known tool in audio signal denoising and source separation [22]. WF denoising algorithm has some disadvantages where the power spectrum of clean audio signal and noisy can not be obtained directly. To overcome these issues, Iterative Wiener Filtering (IWF) algorithm is employed. IWF estimates power spectrum of the original and noisy signal individually using LPC approach [23].

The noisy signal can be represented:

$$y(n) = x(n) + d(n) \tag{18}$$

Where *x(n)* represents as clean signal and *d(n)* is characterized as additive white noise that is Gaussian distributed with zero mean and variance [23]. IWF algorithm is based on autoregressive process which effectively estimates clean audio signal from noisy signal by iteratively employing non-causal WF method [24]. Resulting WF can be written as:

$$H(w) = \frac{P_{xx}(w)}{P_{xx}(w) + P_{dd}(w)} \tag{19}$$

From equation (18) and (19) the filter can be written as:

$$X(w) = H(w)Y(w) \tag{20}$$

Where *w* defined the frequency indices for *X(w)*, *Y(w)* and *H(w)* and these functions are discrete Fourier transform of clean audio signal, noisy audio signal and Wiener filter, consequently. P_{xx}

(w) and P_{dd} *(w)* are power spectral of *x(w)* and *d(w).* After that P_{xx} *(w)* in IFW algorithm is computed as follows:

$$P_{xx}(w) = \frac{g^2}{\left|1 - \sum_{k=1}^{p} a_k e^{-jwk}\right|^2} \tag{21}$$

Where *P* represents as the number of all pole coefficient a_k that can be achieved by LPC analysis. Then the gain of the system *g* is derived by:

$$g^2 = \frac{\frac{2\pi}{N}\sum_{n=0}^{N-1} y^2(n) - 2\pi\sigma^2{}_d}{\int_{-\infty}^{\infty} \frac{1}{\left|1 - \sum_{i=1}^{p} a_i e^{-jkw}\right|^2}} \tag{22}$$

The accuracy of power spectral estimation of noise signal and clean signal relays on the performance of IWF algorithm.

3.4 Block Thresholding Algorithm

Time-frequency block thresholding algorithm was initially introduced by Cai and Silverman [25-28]. The state-of- the art non-diagonal block thresholding audio denoising procedure with further rigorous mathematical treatment of this topic can be found [26]. This algorithm reduces the Short Time Fourier Transform (STFT) coefficient of noisy signal by blocks through identical attenuation factor over each block by using Stein Unbiased Risk Estimator (SURE) theorem [26]. In order to compute the signal attenuation factor over separate time-frequency block. The time-frequency block thresholding estimator stabilizes power subtraction estimation. Then time

frequency plane *{l,k}* is segmented in *I* and B_i. In this case estimator *f* is computed from the noisy data *y* with a constant attenuation factor a_i over each block B_i.

$$\hat{f}[n] = \sum_{i=1}^{I} \sum_{l,k} a_i Y[l,k] g(l.k)[n] \tag{23}$$

To clearly understand how to calculate every a_i, from the risk 'r', to frame energy conversion:

$$\left[r = E\left\{ \left\| f - \hat{f} \right\|^2 \right\} \right] \leq \frac{1}{A} \sum_{i=1}^{I} \sum_{(l,k) \notin Bk} E\left\{ a_i, Y[l,k] - F[l,k]^2 \right\} \tag{24}$$

Since we know that $Y[l,k]=F[l,k]+\varepsilon[l,k]$. Hence, we can demonstrate that upper bound is reduced by selection:

$$a_i = 1 - \frac{1}{\xi_i + 1} \tag{25}$$

Where $\xi_i = F_i^2/\sigma_i^2$ is represented as average a priori SNR in B_i. Then it is calculated directly from equation (24) and (25):

$$F_i^2 = \frac{1}{B_i^{\neq}} \sum_{(l,k) \in Bi} \left| F[l,k]^2 \right| \tag{26}$$

$$\sigma_i^2 = \frac{1}{B_i^{\neq}} \sum_{(l,k) \in Bi} \left| \sigma^2[l,k] \right| \tag{27}$$

Where the noise energy and average signal energy is $B_{i\#}$ and B_i. This block thresholding estimator estimates SNR over each B_i by an average of the noisy signal energy.

$$Y_i^2 = \frac{1}{B_i^{\neq}} \sum_{(l,k)\in Bi} \left| Y[l,k]^2 \right| \tag{28}$$

If $\sigma[l,k]=\sigma_i$ for all $(l,k) \in B_i$ then $\hat{\xi}_i$ is an unbiased estimator of ξ_i. From a power subtraction estimator, the resulting attenuation factor a_i is measured.

$$a_i = 1 - \frac{\lambda}{\xi_i + 1_+} \tag{29}$$

Hence, the Block-thresholding technique can be inferred as a non-diagonal estimator which is resulting from averaged SNR estimations over each block. From all co-efficient in each block, each attenuation factor is calculated and which is normalizes the time-frequency coefficient estimations:

3.5 MFCC Feature Extraction

MFC analysis has been a popular signal representation method used in many audio classification tasks, especially in speech recognition systems [30-31]. Obtaining the MFCCs involves processing of the acoustic signal according to the following steps [32]:

- Divide the signal into frames and apply the Hamming window function.
- Get the amplitude spectrum of each frame.
- Take the log of these spectrum of each frame.
- Convert the mel-frequency scale using triangular shaped filters.
- Apply the discrete cosine transform:

$$C_n = \sqrt{\frac{2}{K}} \sum_{k=1}^{K} (\log S_k) \cos\left[\frac{n(k-0.5)\pi}{K}\right] \qquad (30)$$

Where K defines the number of band-pass filters, S_k defines the Mel-weighted spectrum after passing *k-th* triangular shaped filter and n=1,2…L is the dimension coefficients. The log spectral coefficients are perceptually weighted by a non-linear map of the Mel-frequency scale, which is derived from audio dataset. In addition, Figure 3.5.1 represents as a system block diagram of MFCC feature extraction process.

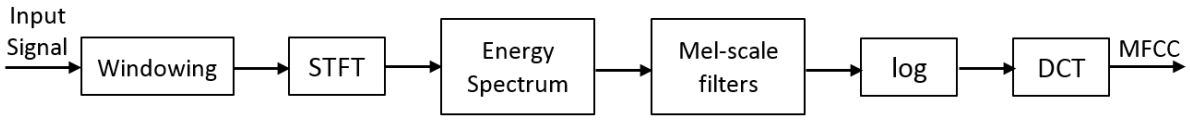


Figure 3.5.1 Block diagram of MFCC feature extraction

The transform formula for the Mel-frequency scale and linear frequency scale is defined as:

$$Mel(f) = 2595 \times \log_{10}\left(1 + \frac{f}{700}\right) \quad (31)$$

Where f represents the frequency Hertz.

3.6 SVM Classification

SVM performs multi-class classification with one-versus-the-rest, pairwise comparison, and multiclass objective functions. During the one-versus-the-rest procedure, classifiers are trained to separate one class from the rest. The multi-class classification is then carried out according to the maximal output of these classifiers. In pairwise comparison, a classifier is trained for each possible pair of classes and the observation in question is assigned to the class getting the highest number of classification "votes" among all the classifiers. In the multi-class objective-function, the objective function of SVM is directly modified to allow the simultaneous computation of a multi-class classifier [34]. SVM attempts to find the hyperplane separating two classes of data that will generalize best to future data [35]. Such a hyperplane is then so called maximum margin hyperplane which maximizes the distance to the closest point from each class. More concretely, given data points $\{X_0,...,X_N\}$ and class labels $\{y_o,..,y_N\}$, y_i $\{-1,1\}$, any hyperplane separating the two data classes has the form

$$y_i\left(w^T X_i + b\right) \rangle 0 \ \textit{for any } i \quad (32)$$

Let $\{w_k\}$ be the set of all such hyperplanes. The maximum margin hyperplane is denoted by

$$w = \sum_{i=0}^{N} \alpha_i y_i X_i \quad (33)$$

Where $\{a_0, a_1, \ldots a_N\}$ maximize,

$$L_D = \sum_{i=0}^{N} \alpha_i - \frac{1}{2} \sum_{i=0}^{N} \sum_{j=0}^{N} \alpha_i \alpha_j y_i y_j X_i T_i^T X_j \quad (34)$$

Corresponding to

$$\sum_{i=0}^{N} \alpha_i y_i = 0 \;\; \alpha_i \geq 0 \text{ for any } i \quad (36)$$

For linearly separable data, only a subset of the *a*, will be non-zero. These points are called the support vectors and all classifications performed by the SVM depend only on these points. SVM systems can accurately identify critical samples that will become the support vectors, training time and labeling effort which, in the best case, can be reduced with no impact on classifier accuracy. Since the data points X only enter calculations via dot products, one can transform them to another feature space via a function F(x). The representation of the data in this feature space need never be explicitly calculated if there is an appropriate Mercer kernel operator for which

$$K(X_i, X_j) = \phi(X_i) \cdot \phi(X_j) \quad (37)$$

Data not linearly separable in the original space may become separable in this feature space. In our implementations, a radial basis classifier kernel

$$K(X_i, X_j) = e^{-\gamma D^2(X_i, X_j)} \quad (38)$$

Thus the space of possible classifier functions consists of linear combinations of weighted Gaussians around key training instances.

3.7 Naive Bayes Classification

In this section we first explain the general form of the NB classification framework. Let $X = \{x_1, \ldots\ldots, x_n\}$ denote the set of local feature vectors obtained from a target audio, we want to find its class C. This can be formulated as a Maximum-a-Posteriori (MAP) problem, where feature set X is assigned to the class that maximizes the posterior probability $p(C|X)$. Assuming equal class prior $p(C)$, the MAP problem then reduces to Maximum Likelihood (ML) estimation of the conditional probability $p(X|C)$ [36-37]. The NB assumption specifies a simple probability model that the feature vectors $x_1 \ldots\ldots x_n$. extracted from each target audio are generated independently from each class C with identical distribution. So that we have the following classification rule,

$$\hat{C} = \arg\max_{C} p(X \mid C) \tag{39}$$

$$= \arg\max_{C} \prod_{j} p(X_j \mid C)$$

$$= \arg\min_{C} -\sum_{j} \log p(X_j \mid C)$$

Hence, the problem is converted to the estimation of $p(x|C)$, the conditional probability of local feature for x_j each class C. Depending on how $p(x_j|C)$ is modeled, different NB classifiers can be defined. Note that Equation-39, describes an extension of the standard NB framework, which is used for the classification of features by assuming each attribute is generated independently given the class label. Here we have generalized the NB framework to deal with the classification of collections of feature vectors.

CHAPTER IV: IMPLEMENTATION

4.1 Software Implementation

The performance analysis of the aforementioned denoising algorithms is initially implemented in the MATLAB software for computer simulation. Our test data set contains four different classes of audio data: animal, bird, human, and vehicle. Each class contains 100 different audio files. The length of each audio file (*.wav files) is 3s and sampled at 8 KHz. Figure. 4.1.1. shows the typical human speech waveform in time-domain and frequency-domain (spectrogram). On left human speech waveform in time domain and right side represents corresponding waveform spectrogram with frequency domain.

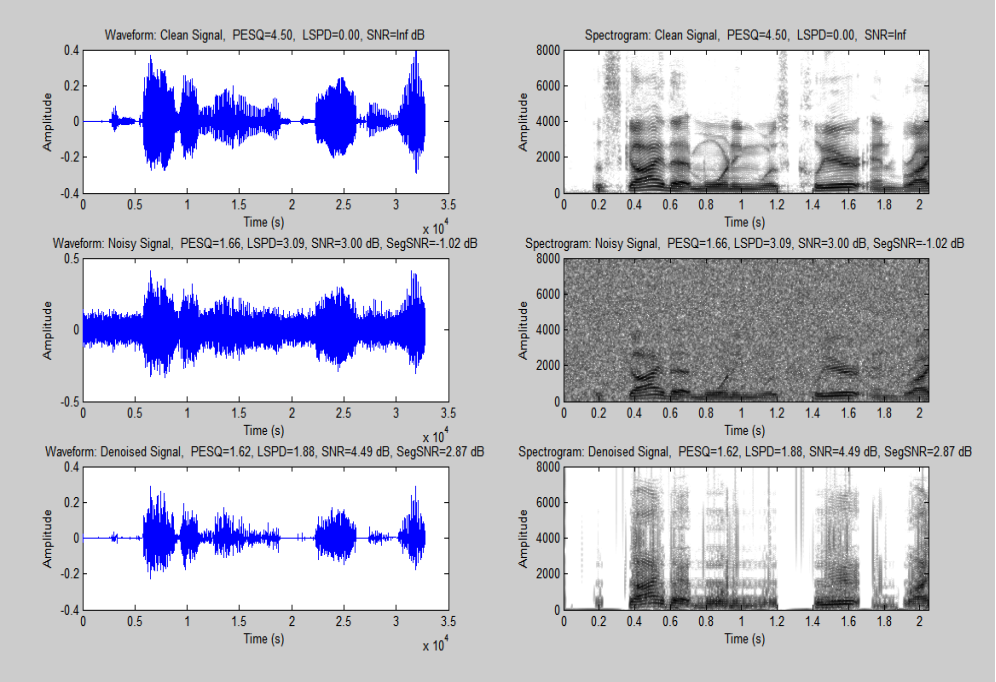


Figure 4.1.1 Human speech waveform in time-domain and frequency-domain

To test the algorithm performance, initially we added AWGN noise to each audio file, and then sequentially added the UAV's propeller and motor noise. Moreover, the noise sources are added to the clean audio signal in six particular SNR levels: (3dB, 6dB, 9dB, 12dB, 15dB, 20dB). After that, the clean audio signal is corrupted by White Gaussian noise or PM or DJI noise with different amplitudes. Then the resulting noisy signals are used for evaluating the performance of denoising algorithms.

LMS: In this implementation, we applied the sum of the noise and the clean signal as the input of the channel. Moreover, equations (3)-(6), summarize the software implementation process of the LMS denoising algorithm. The rate of convergence of LMS algorithm is mainly controlled by step size μ. This factor determines the rate of convergence of the filter. We selected the filter length as 30 samples, and number of samples delay typically as 200 for single input LMS implementation, and the step size typically as 0.017. An initial output filter is calculated from the input response and an initial weighting coefficient. The error signal, which is the difference between the output and desired signal, is estimated. Then, the weighting coefficient vector is updated using the previous weighting coefficient, step factor, error, and input response. Then, we employed performance measure metrics to evaluate the denoised signal.

DWT: To determine the SNR of the target audio signal, initially we added noise sources (AWGN or PM or DJI) to the clean audio signal to produce a noisy signal. After that, we took this noisy signal as an input to the DWT algorithm. Then, we applied thresholding to estimate the noise level. We used the soft thresholding approach. After that, we applied inverse DWT to reconstruct the signal. Then, we obtained the denoised signal. Finally, we employed performance measure metrics to evaluate the performance of this implemented algorithm. We used Daubechies (db)

wavelet as a mother wavelet function. To select the optimum decomposition level, an experiment is done to evaluate the performance of the wavelet transform with different levels of decomposition. We observed that at 6 level of decomposition results in the best performance, and that is explicitly demonstrated in Fig. 4.1.2.

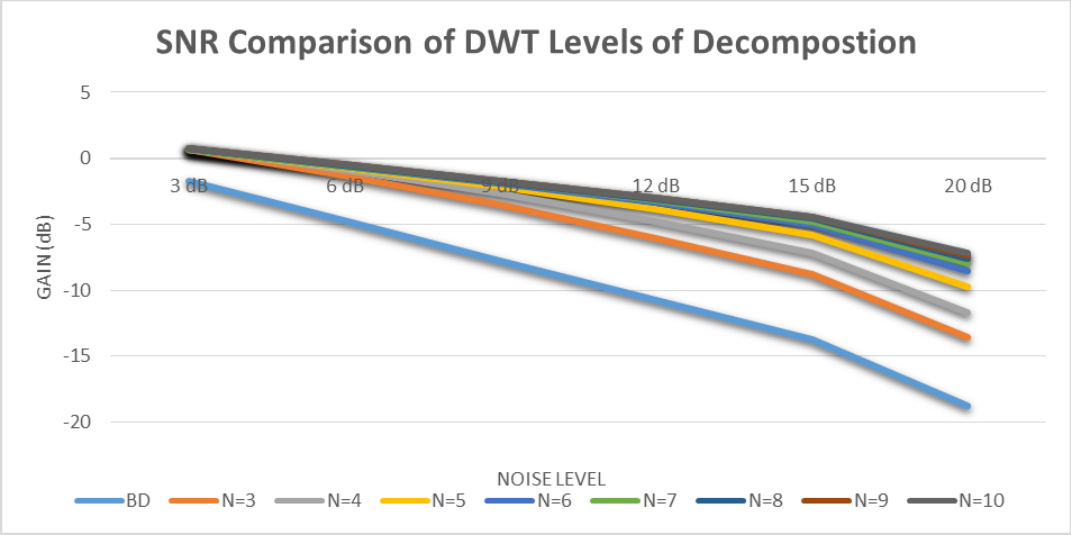


Figure 4.1.2 Levels of decomposition

BTH: To implement this algorithm, we summed noise sources with the clean signal to generate the noisy audio signal. STFT with half-overlapping windows were used. After that, we employed a non-diagonal BTH estimation algorithm. Optimum block sizes are calculated by minimizing the estimation risk. We computed the threshold level λ as 4.7 and block sizes were 4, 8, 16, 32, etc. Then, a non-diagonal BTH estimator is derived from averaged SNR estimation over blocks. The implementation of the BTH algorithm is summarized by eqns. (18)-(24). Similarly, we applied other performance measure metrics to evaluate this algorithm.

To obtain a better denoised signal, we applied a noisy audio signal as the input of the WF algorithm, and then used LPC analysis. The IWF algorithm is implemented as summarized by

eqns. (13)-(17). The smoothing factor G=0.9 and margin is set 0.01. The IWF process is repeated in order to obtain a cleaner audio signal. Thus, this algorithm operates in the short-time segment of the audio signal due to the non-stationary nature of the target audio signal.

Simulation results of the implemented denoising algorithms are explained in detail in the software results section of Chapter V.

4.2 Hardware Implementation

For these experiments, the Texas Instruments (TI) TMS320C5535 was chosen as the hardware testing platform of our project. This is because of TI's prominence in the development and production of Digital Signal Processors (DSP), and in education/training. It is a high performance device with rich features that is widely used in industry. The C5535 eZdsp is an evaluation tool for the TI TMS320C535 DSP series [43]. The block diagram of C5535 eZdsp is shown in Figure. 4.2.1.

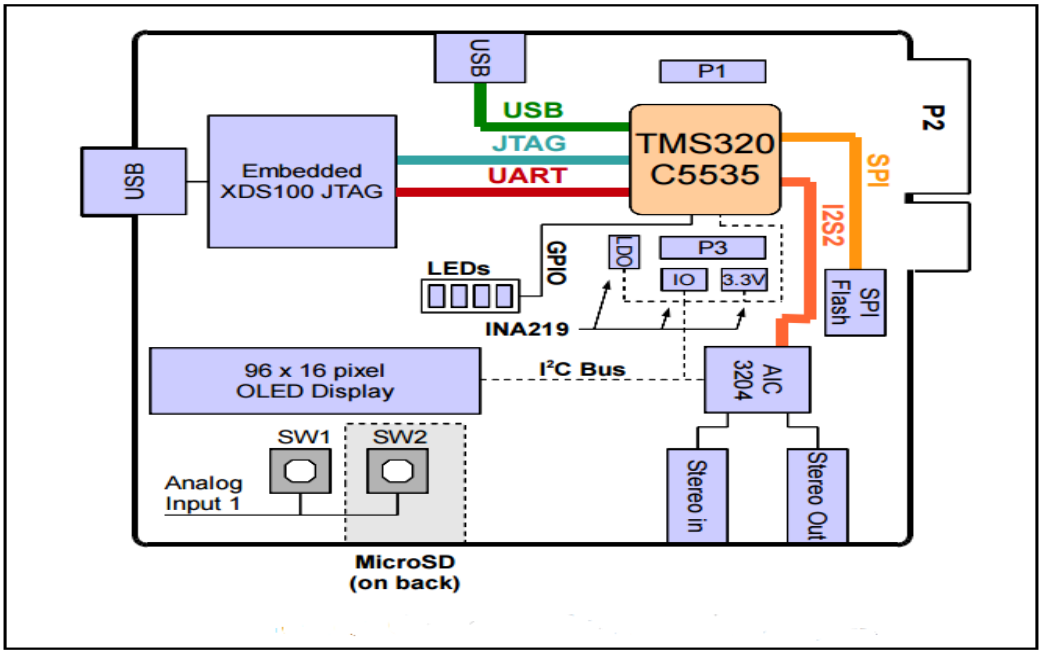


Figure 4.2.1 Block diagram of the TMS320C5535 DSP board

The USB bus powered tool allows the user to evaluate the TMS320C5535 DSP with the TLV320AIC3204 codec and the Code Composer Studio (CSS) IDETM software development tools. Figure. 4.2.2 and 4.2.3. represent the key features of the TMS320C5535 DSP board on top and bottom view.

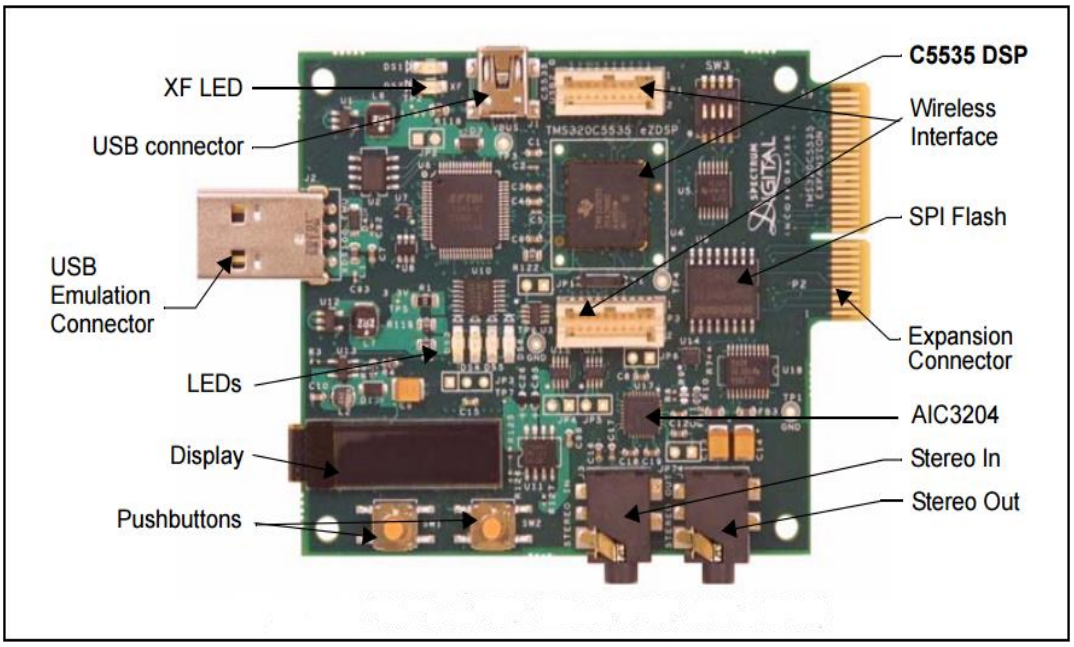


Figure 4.2.2 Key Features of the TMS320C5535 DSP board on top view

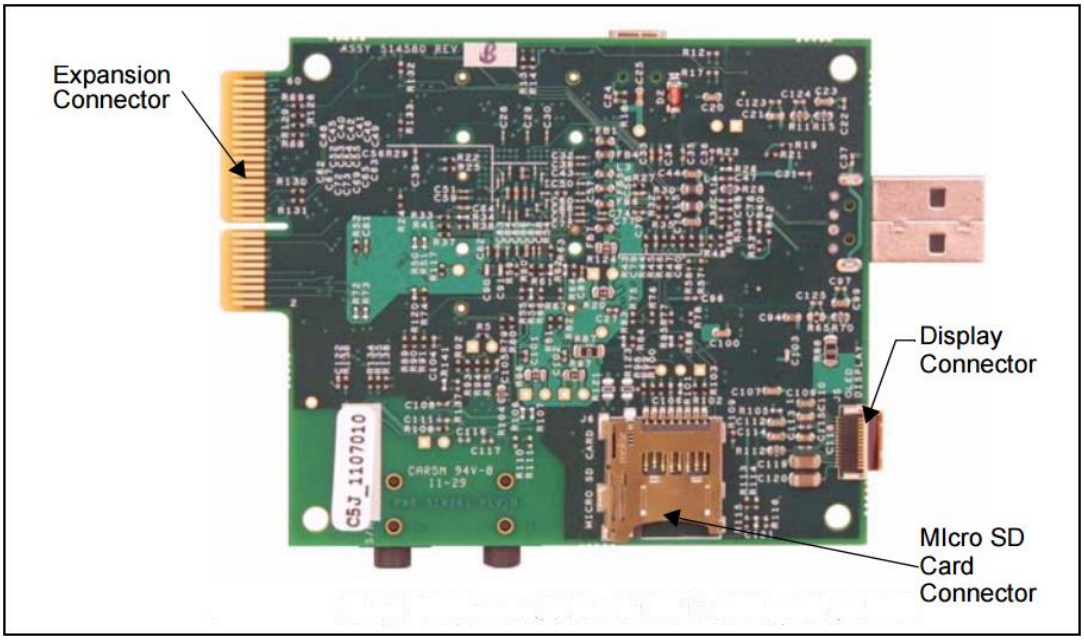


Figure 4.2.3 Key Features of the TMS320C5535 DSP board on bottom view

The C5535 eZdsp has the following features:

- Texas Instrument's TMS320C5535 Digital Signal Processor
- Texas Instruments TLV320AIC3204 Stereo Codec (stereo in, stereo out)
- Micro SD card connector
- USB 2.0 interface to C5535 processor
- 8 Mbytes SPI flash
- 2 user readable push button switches
- Embedded USB XDS100 JTAG emulator
- I2C OLED display
- Compatible with Texas Instruments Code Composer Studio v4
- USB cable.
- Power provided by USB interface.

For this work, the system hardware and software are introduced, and the tasks of the project are detailed as well. How the system is implemented, and how it works are explained. The initial step is to properly connect the system hardware and then set up the communication interface via USB cable, and necessary wires between the hardware and computer. Next, we employ individually LMS and DWT denoising algorithms through CSS [44]. When the system is running, the noisy audio signal is passing to the C5535 DSP board via the audio line in port, then the signal is processed with a denoising based algorithm based on our predefined denoising method. The RTDX (real time data exchange) function of the CCS allows the C5535 DSP chip to interconnect and exchange information with the PC via a USB interface. Therefore, the DSP chip could directly exchange information with PC without discontinuing the execution of the

current job. Figure. 4.2.4. Presents the complete hardware experimental setup in laboratory. The resulting signal after denoising is passed out through the audio line out port from the C5535 DSP board. It is then captured and further analyzed in audacity software to measures its optimum performance. The performance of the denoised signal is explained in detail in the hardware results section of chapter V.

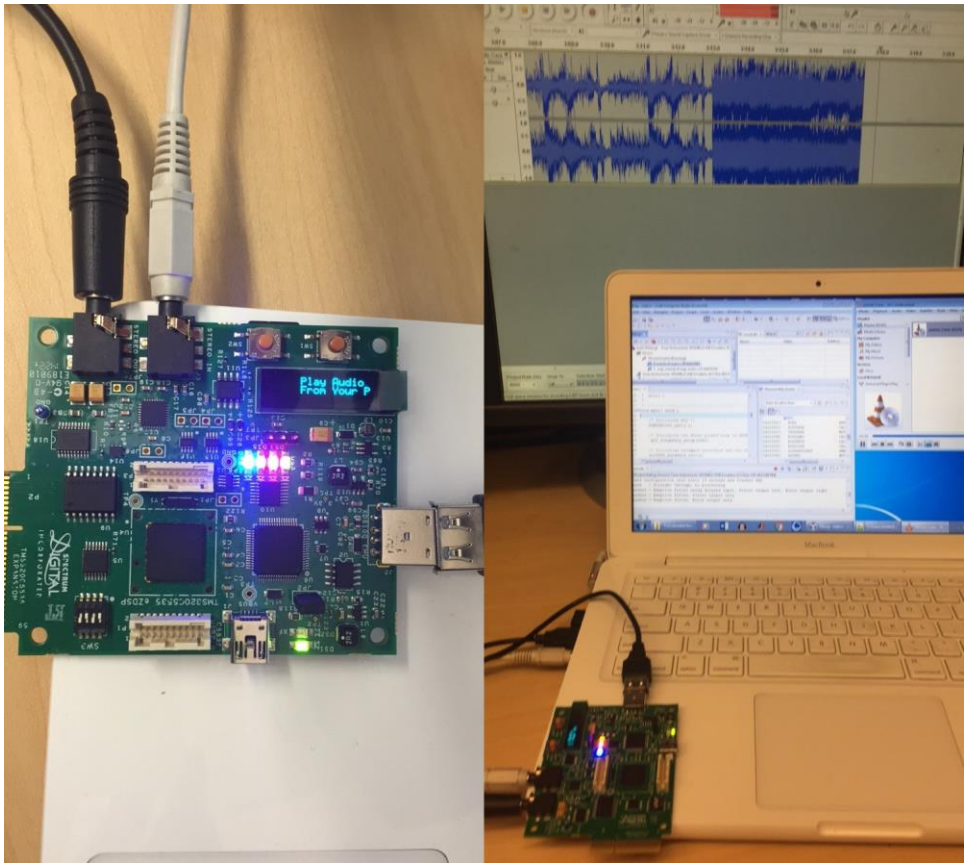


Figure 4.2.4 Hardware experimental platform in laboratory

4.3 Classification Implementation

Waikato Environment of Knowledge Analysis (WEKA) toolkit is used to implement the SVM and Naive Bayes classifier. It contains modules like data preprocessing, classification, clustering and association rule extraction. It has 49 data preprocessing tools, 76 classification algorithms, and 3 graphical user interface and algorithms for association rules [45]. It contains different types of data formats namely Attribute Relation File Format (AREF), and Comma Separated Values (CSV). SVM and NB are very well suited for this problem because of the small number of classes and huge amount of training data per class. Initially we employed SVM with a linear kernel and complexity 1.0. Choosing the kernel function is the trickiest part of using SVM. The kernel function is important because it creates the kernel matrix, which summarizes all the data. Many principles have been proposed. However, in this research we employed a low degree polynomial kernel. They are trained with the sequential minimal optimization (SMO) algorithm using the windowed training data. Fig. 4.3.1 Illustrates the parameter setting for SVM classifier in software platform. The buildLogisticModels option determines whether or not to fit logistic models to the outputs for proper probability estimates. We set it to the True condition in our case. The number of folds for cross validation are used to generate training data for logistic models. C is defined as a complexity parameter, and it is the upper bound of alpha's. The filter type determines, how and if the data will be transformed, so we decide to use normalize training data in filter type. However, we do not change the rest of the parameters including epsilon, checks turnoff, and toleranceParamer, etc. We kept them at default settings. After setting all the parameters properly, we then inserted our .arff training and testing data and then run SVM in

software. Figure. 4.3.2. and Figure. 4.3.3 explain the necessary steps for complete implementation procedures of SVM classifier in software platform.

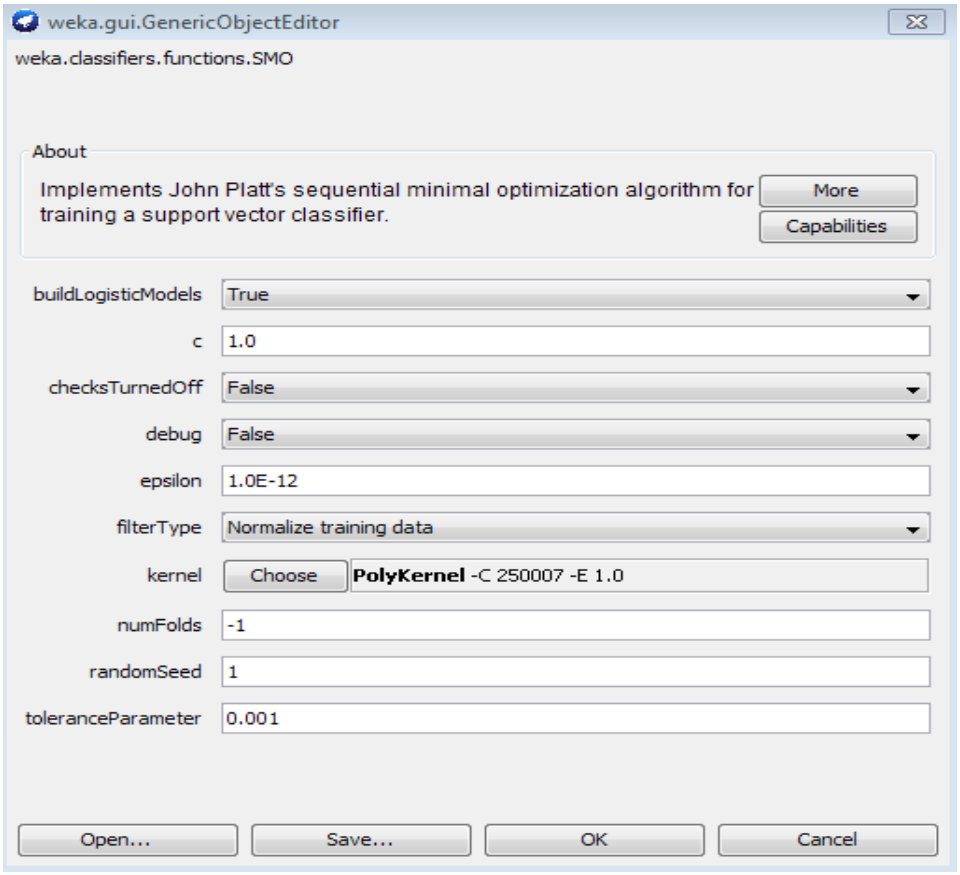


Figure 4.3.1 SVM/SMO parameter selection

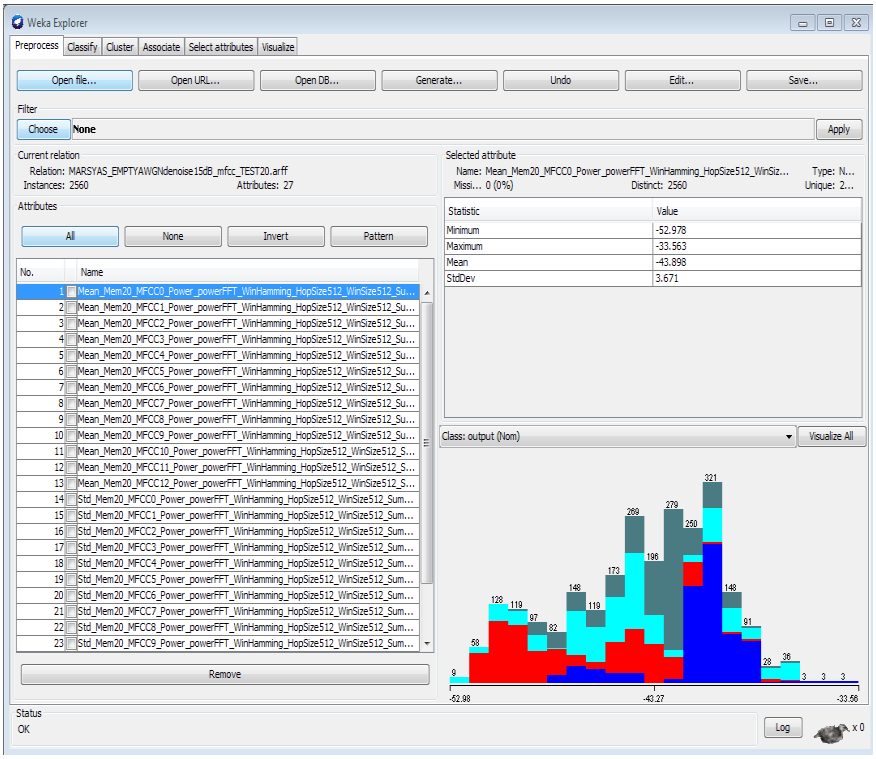


Figure 4.3.2 SVM/SMO parameter selection

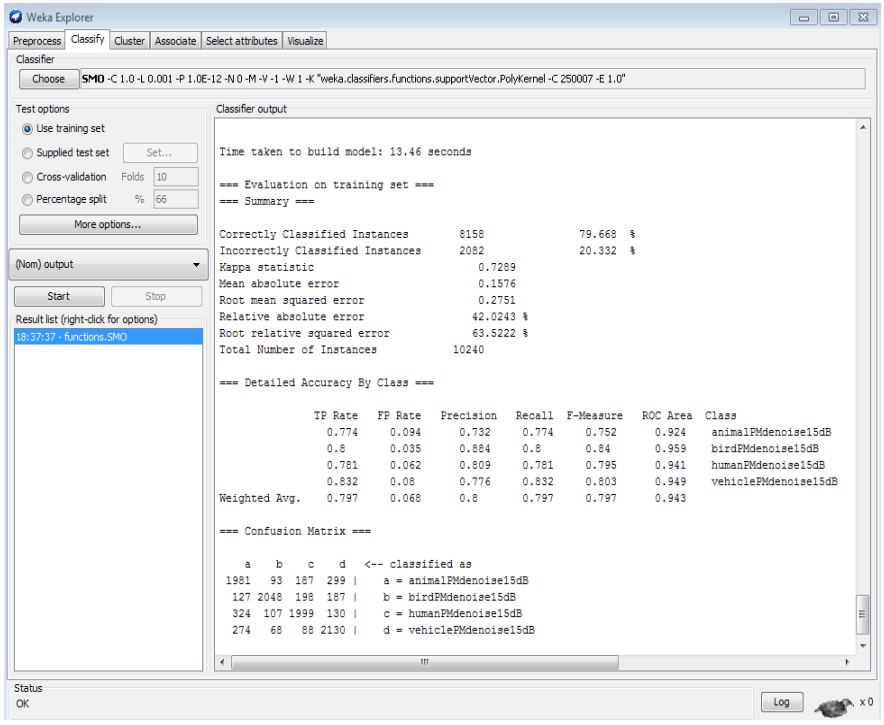


Figure 4.3.3 SVM/SMO evaluation of training set

For comparison, a Naive Bayes classifier is implemented and tested. The Naive Bayes method provides probabilistic outputs. Figure. 4.3.4 illustrates the parameter setting for Naive Bayes classifier in software platform. We set almost every parameters by default option.This means that Naive Bayes models can assess the value of the probability (varying from 0 to 1) that a given compound can be predicted as active. By moving the threshold from 0 to 1, and imposing that a compound can be predicted as active, if the corresponding probability exceeds the current threshold, one can build the ROC (Receiver Operating Characteristic) curve. We set almost every parameter to the default option.

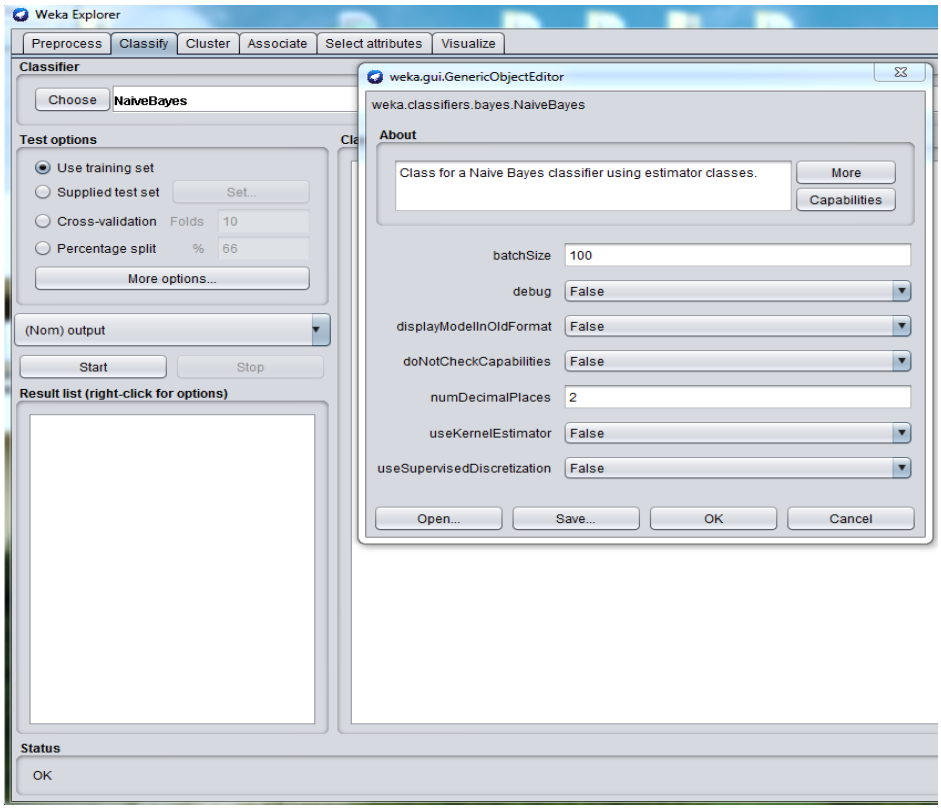


Figure 4.3.4 Naive Bayes classifier parameter selection in software platform

Moreover, Figure. 4.3.5. and 4.3.6., illustrates the ROC curve characteristic for noise and denoisied animal wave file in AWGN noise condition with SVM and NB classifier.

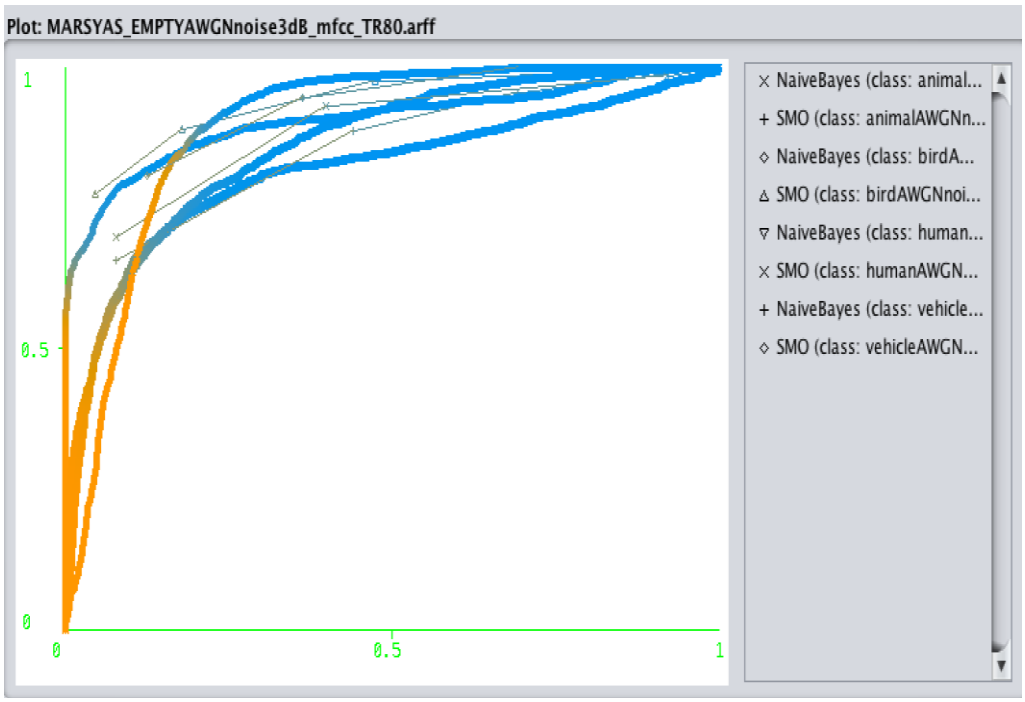


Figure 4.3.5 ROC curve of animal AWGN noise scenario with SVM and NB classifier

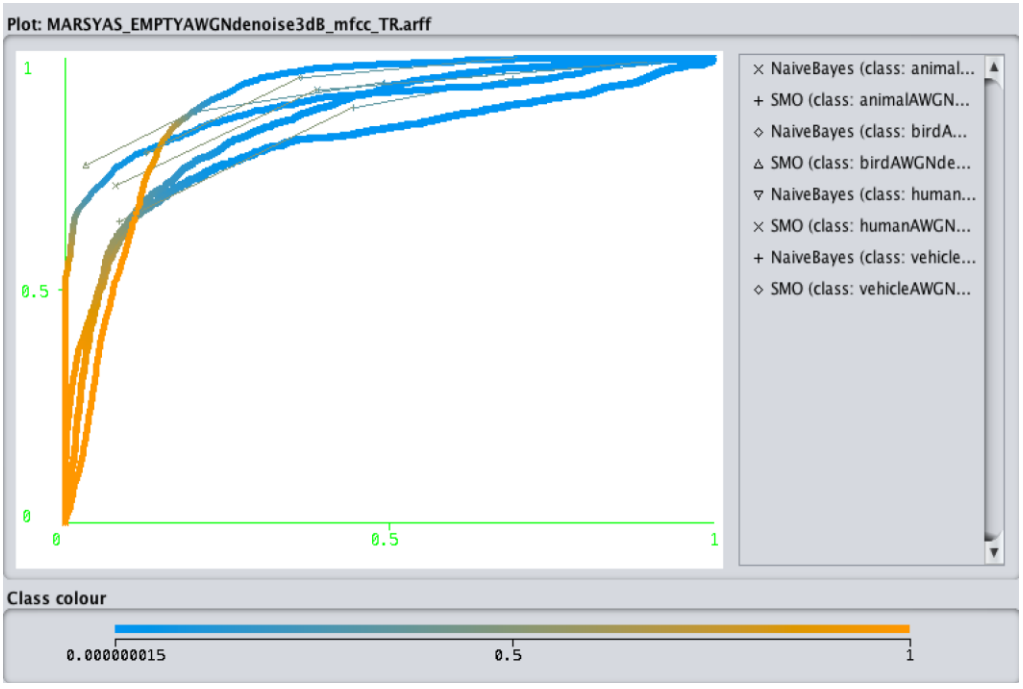


Figure 4.3.6 ROC curve of animal, denoised scenario with SVM and NB classifier

Evaluation of classification is a very essential since results and conclusions are always based on the evaluation so that all the results are grouped into the many sub items. The classification in the table for correctly and incorrectly classified instances and mean absolute error will be partitioned in percentage value and subsequently. Appendix B represents complete classification results data. Simulation results of above implemented classification is explained details in results section in Appendix C.

CHAPTER V: RESULTS AND DISCUSSION

5.1 Simulation Results

In this section, the resulting computer simulations that verify the optimum performance of the implemented denoising algorithms are presented. Initially, we evaluated the denoising algorithms performance with four different classes. Figure. 5.1.1 depicts and compares the algorithms' performance for four different audio classes with respect to the SNR metric.

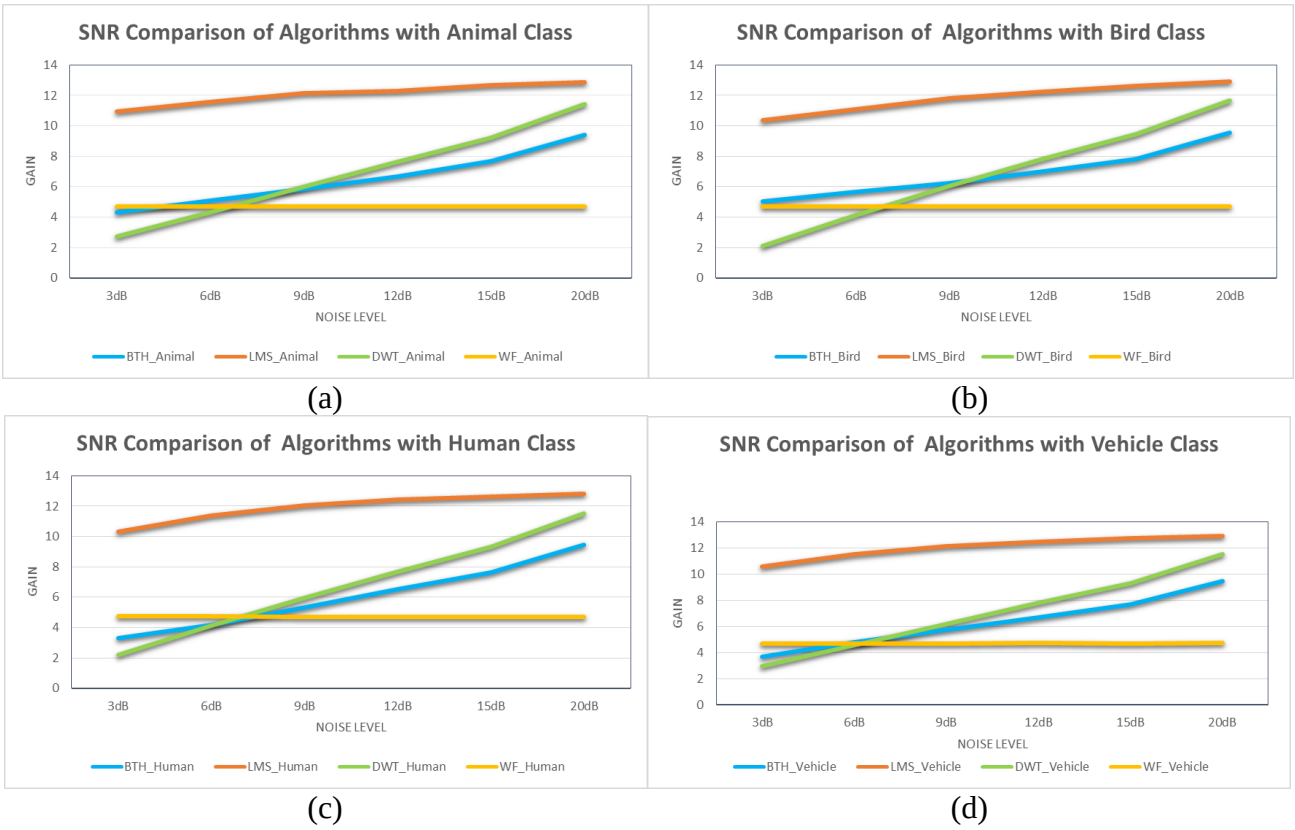


Figure 5.1.1 Denoising algorithms comparison with four audio classes in SNR scenario:
(a) Animal, (b) Bird, (c) Human and (d) Vehicle.

From Figure. 5.1.1. one can observe that the LMS algorithm performance is superior to DWT, BTH, and WF in all class scenarios. SNR gain level reached up to 13 dB, when noise level 20dB. The greater its value, the cleaner signal is, and the better the sound quality is. In addition, both the DWT

and BTH algorithm also linearly increased as the noise level increased. However, WF did not offer good performance in this analysis. Similarly, we evaluated the algorithms' performance in the SSNR case, in order to more rigorously analyze the performance. Figure. 5.1.2 illustrates and compares the algorithms' performance with all target audio classes in the SSNR aspect. It calculates the SSNR value every 20ms from the denoised signal.

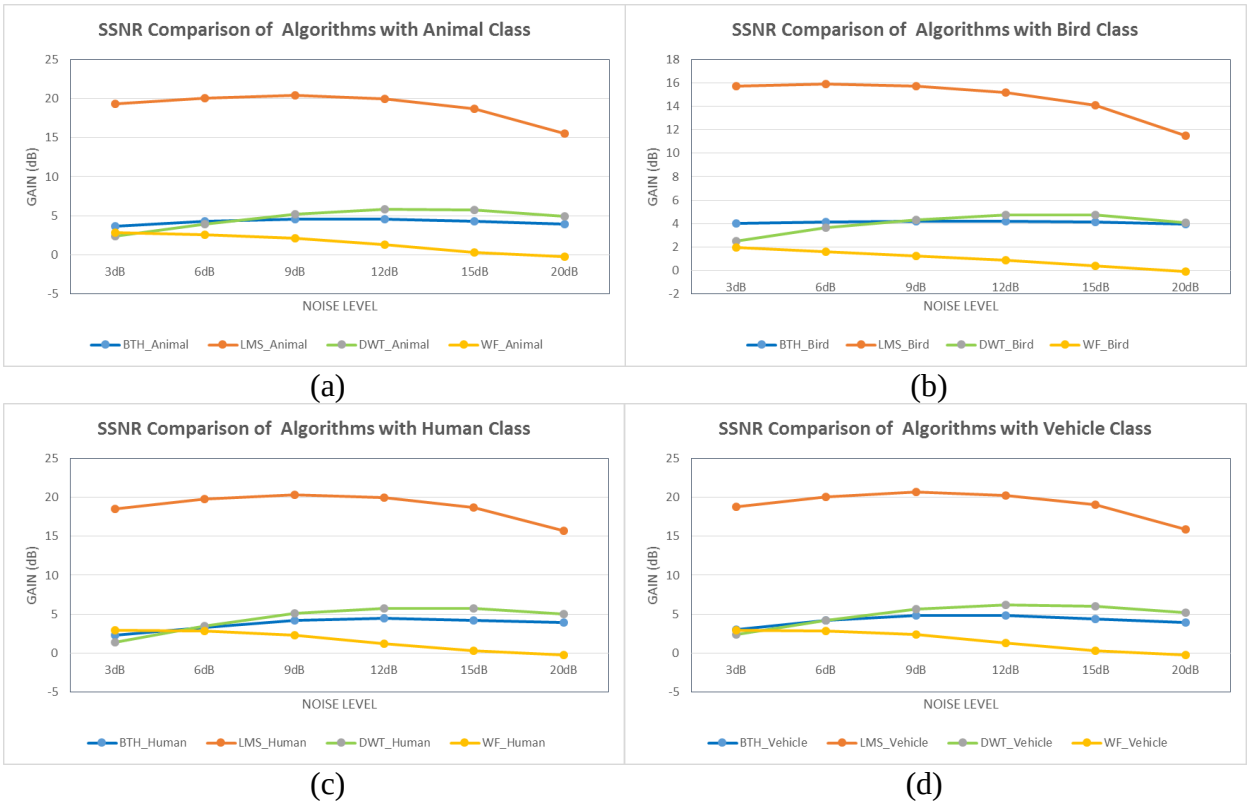


Figure 5.1.2 Denoising algorithms comparison with four audio classes in SSNR scenario:
(a) Animal, (b) Bird, (c) Human and (d) Vehicle.

From the above graph we noticed that LMS is better compared to DWT, BTH and WF. We noticed that its SSNR gain increased as the noise level increased but after a 9 dB level, it started decreasing. Moreover, DWT and BTH also increased, but after 12 dB level of noise, both algorithms' SSNR gain decreased. WF increased a little but it mostly offered a negative gain, meaning this did not provide a

good denoised signal. Figure. 5.1.3 compares the denoising algorithms' performance with respect to AWGN, Propeller and Motor (PM) noise, and DJI drone noise in SNR metrics.

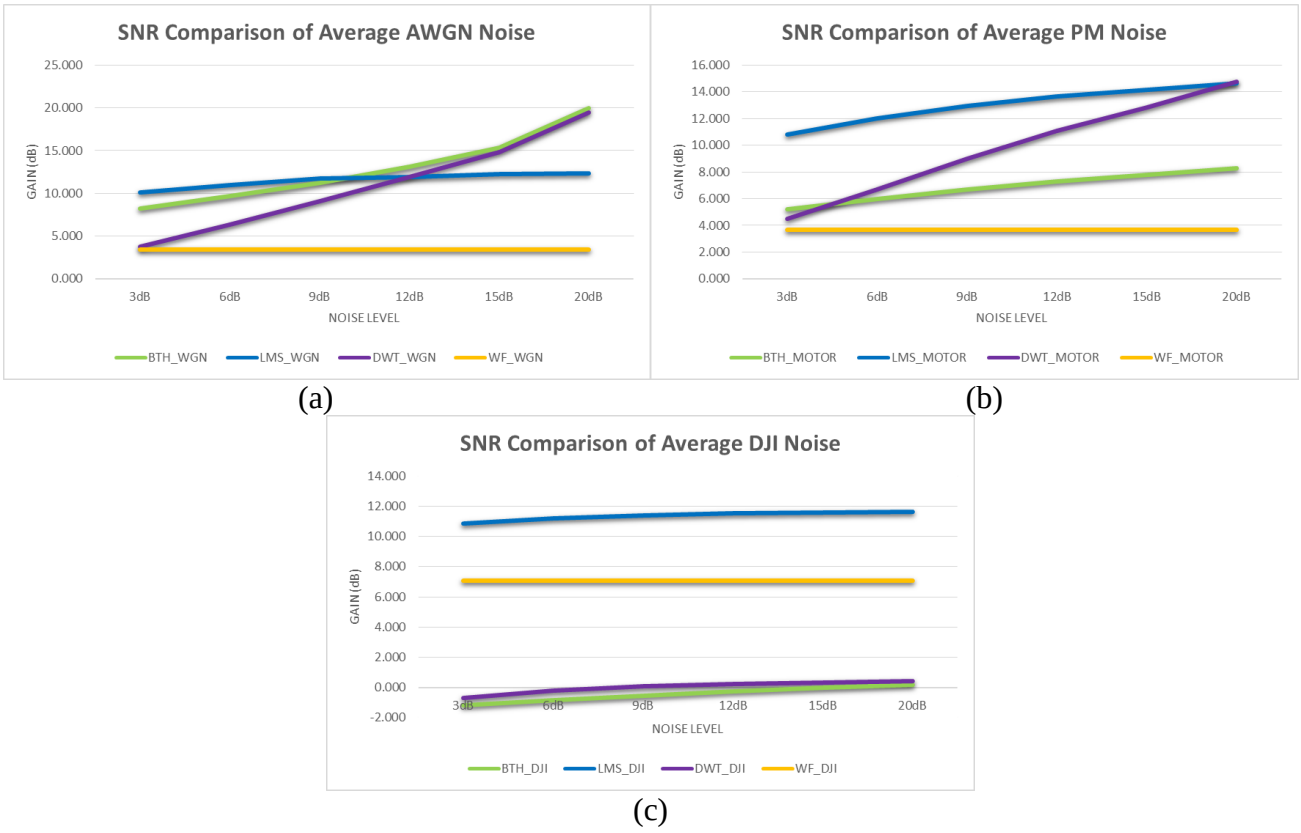


Figure 5.1.3 Denoising algorithm comparison with respect to noise with SNR scenario:
(a) AWGN Noise, (b) Propeller and Motor (PM) Noise, (c) DJI Drone Noise.

From the above graph, we observed that the LMS performance in SNR improvement is higher than DWT, BTH and WF except for AWGN noise. In the AWGN noise case, the DWT and BTH algorithms' SNR gain increased linearly as the noise level increased. BTH performed better than DWT in AWGN, but in PM noise, DWT offered improved SNR compare to BTH. Thus, in these two noise scenarios, WF did not offer optimum performance compared to the other algorithms but had consistent performance in all noise cases. In the DJI drone noise scenario, WF had better SNR values compared to the DWT and BTH algorithms. The SNR gain reached up to 7 indicating a constant

level. The DJI drone noise scenario is the one with the strongest noise (most difficult to remove). This is supported by observing that the SNR gains for BTH and DWT algorithms drastically falls to 0 dB in the DJI noise case. Figure. 5.1.4 compares the denoising algorithms' performance with respect to AWGN, Propeller and Motor (PM) noise, and DJI drone noise in SSNR characteristic

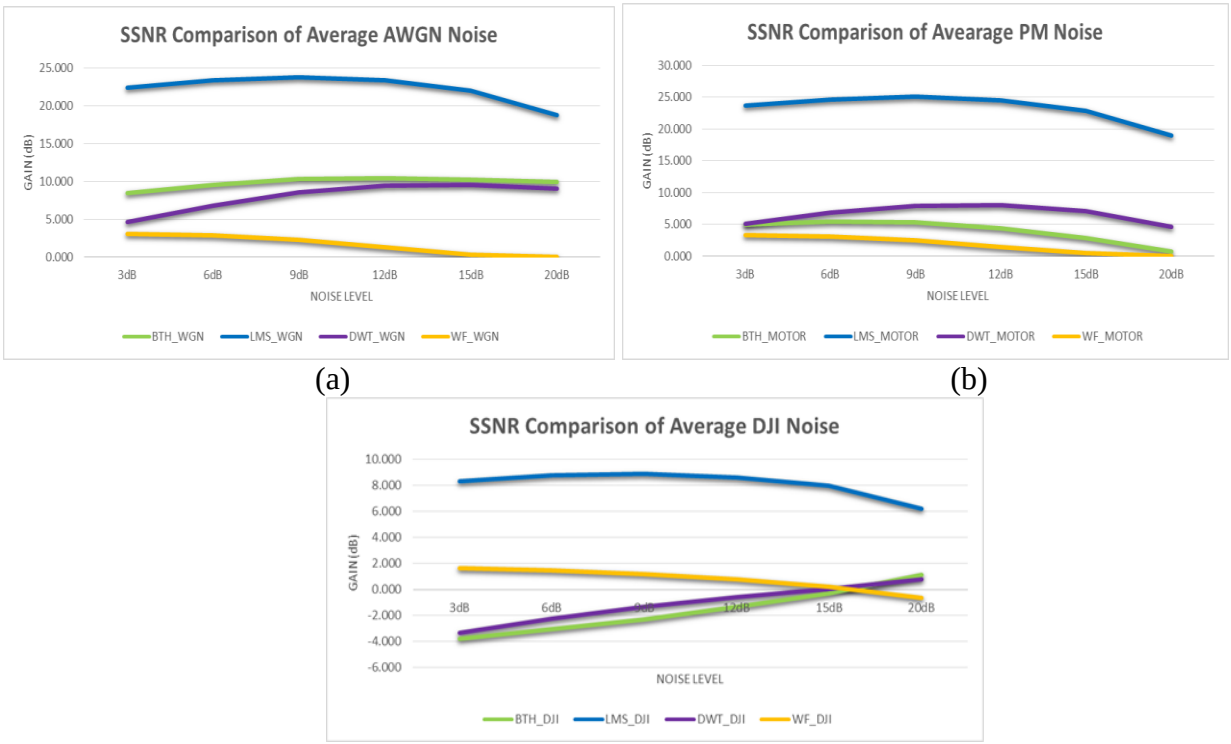


Figure 5.1.4 Denoising algorithm comparison with respect to noise with SSNR scenario:
(a) AWGN Noise, (b) Propeller and Motor (PM) Noise, (c) DJI Drone Noise

From the above graph, we see that LMS is better for SNR improvement compared to DWT, BTH, and WF. It's important to notice that in all three noise cases, the LMS algorithm's SNR value started decreasing when the noise level reached 9 dB. Similarly, we saw that in AWGN noise, BTH performed better than DWT, but in PM noise, DWT achieved better SNR values compared to BTH and WF. Hence, in these two noise scenarios, WF did not offer optimum performance compared to the other algorithms. But in the DJI drone noise scenario, surprisingly, WF gave a higher SNR value

compared to the DWT and BTH algorithms. It again confirms that in DJI drone noise scenario, it is hard to eliminate the noise compared to the AWGN and PM noise cases. When the noise level is 3dB, both DWT and BTH offered negative SNR gain, but these gain improve slightly as the noise level increased to 20 dB. Figure. 5.1.5 demonstrates the overall performance of our implemented denoising algorithm with respect to SNR metric.

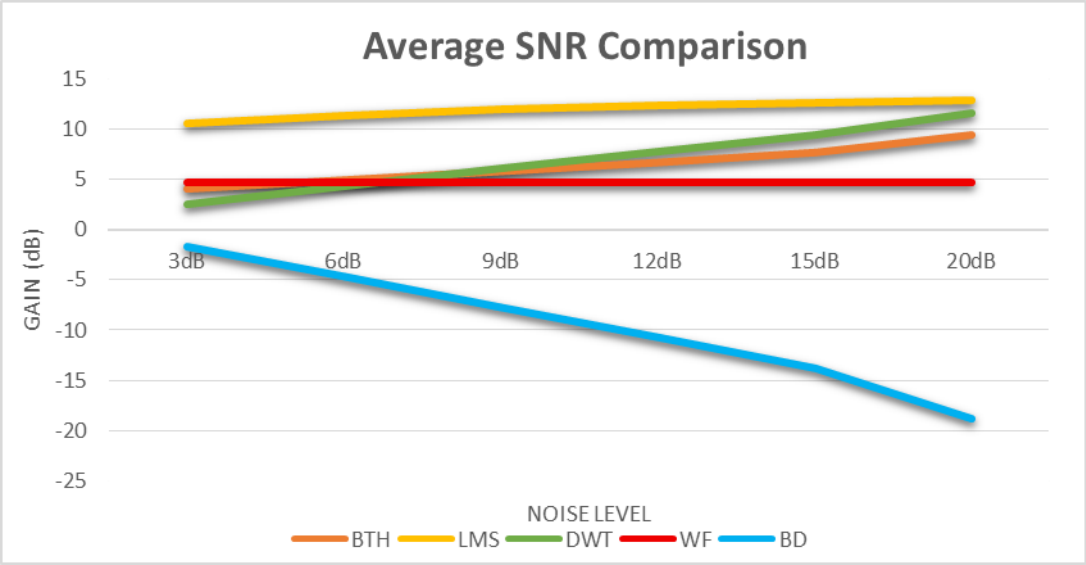


Figure 5.1.5 Overall SNR comparison within denoising algorithms

From Figure. 5.1.5, it is obvious that the adaptive filter model based LMS algorithm is more suitable for denoising signals and raising the SNR gain as compared to the DWT, BTH and WF algorithms. Before denoising (BD) the signals, one can notice that the SNR gain decreases as the noise level increases. But after employing a denoising algorithm, we obtain a denoised signal and a higher SNR gain, clearly indicated in the graph.

Figure. 5.1.6 demonstrates the overall performance of our implemented denoising algorithms with respect to SSNR metric.

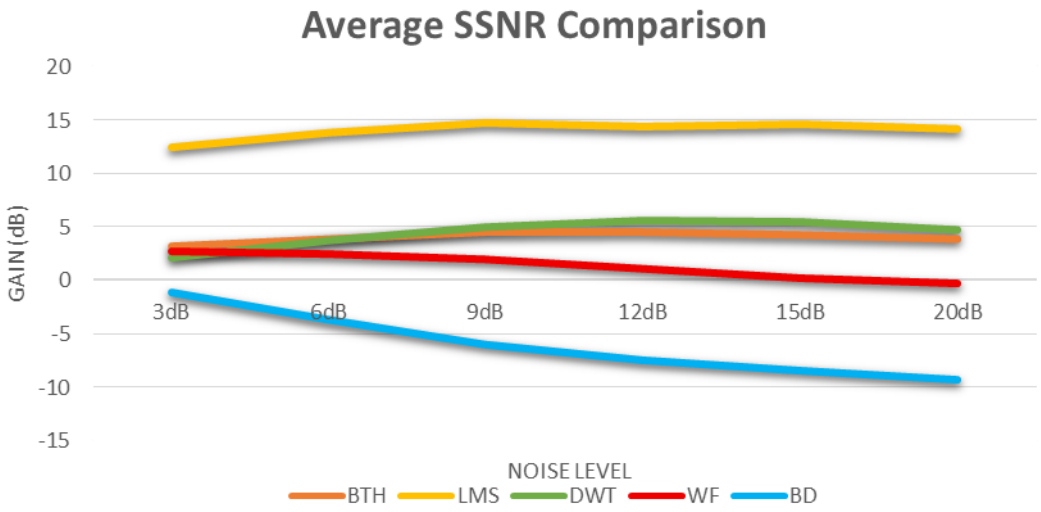


Figure 5.1.6 Overall SSNR comparison within denoising algorithms

A significant improvement of SSNR gains with the LMS algorithm is observed from the above graphical representation. The SSNR gain of 15 dB can be reached. Figures. 5.1.7 and 5.1.8 explain the overall performance of our implemented denoising algorithm with respect to LLR and LSPD metrics

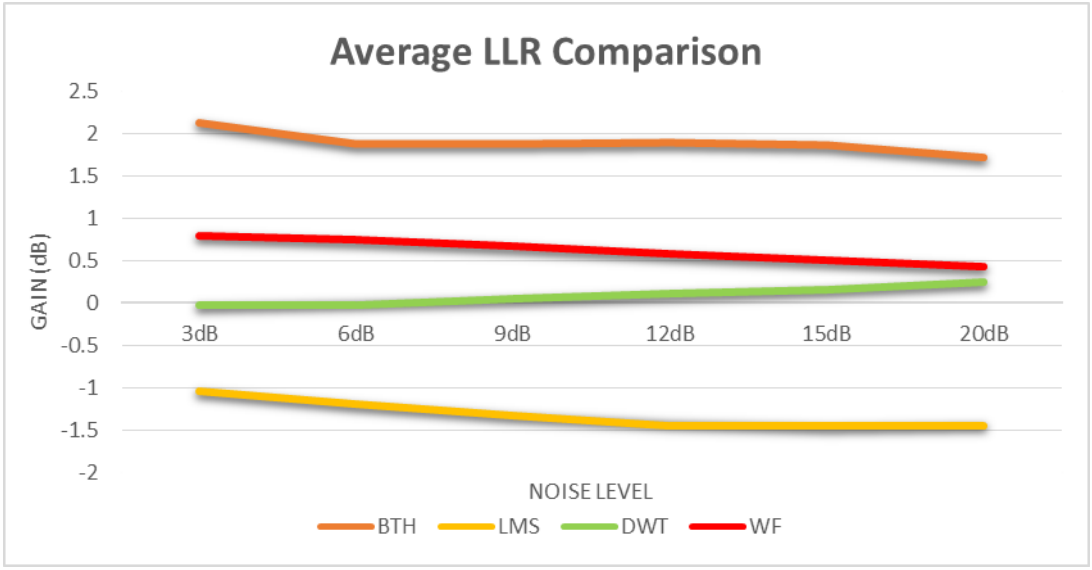


Figure 5.1.7 Overall LLR comparison within denoising algorithms

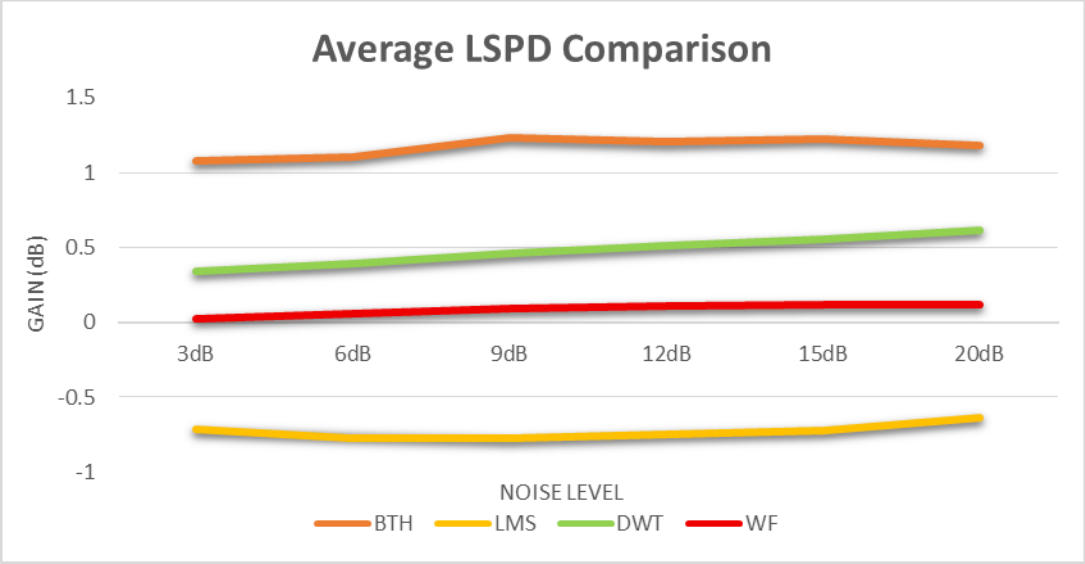


Figure 5.1.8 Overall LSPD comparison within denoising algorithms

From the above two figures, LMS has the lowest gains in both LLR and LSPD. Unlike SNR and the SSNR, the lower gain in LLR and LSPD, the higher performance the algorithm has define. Therefore we summarize that the performance of the LMS algorithm is better than the other implemented algorithms.

5.2 Hardware Test Results

Based on the results of the software implementations of the previous section, we concluded that the adaptive LMS and DWT algorithms' performance is superior compared to the rest. Thus, we implement these two algorithms in the C5535 DSP board for hardware testing. Figure. 5.2.1 compares the algorithms' performance with four different audio classes in SNR metric.

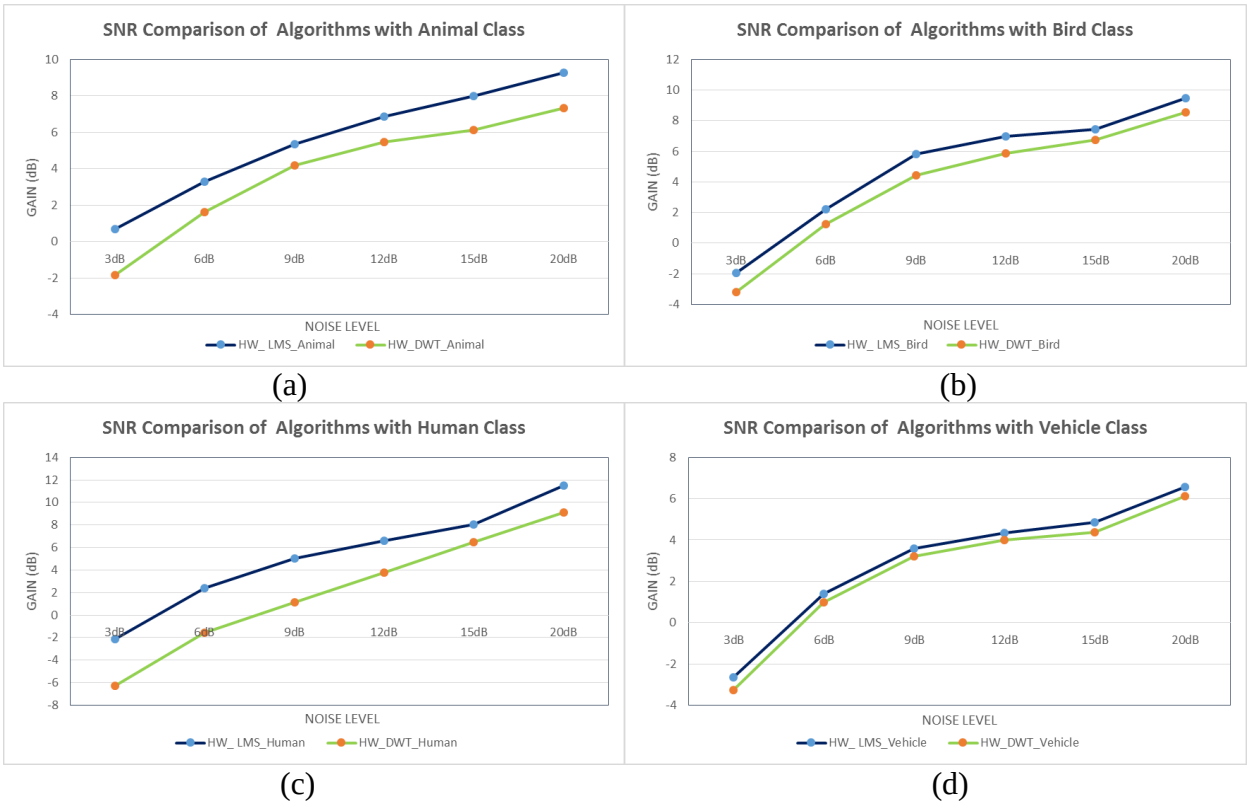


Figure 5.2.1 Hardware SNR comparison within denoising algorithms with four audio classes: (a) Animal, (b) Bird, (c) Human and (d) Vehicle

As can be seen from Figure. 5.2.1, the adaptive LMS algorithm's performance is marginal superior compared to the DWT algorithm. Here, the achieved SNR level reaches up to 9 dB for the LMS algorithm for the animal and bird classes, but in the human class, it offers an even higher SNR gain at 20 dB noise level. Both algorithms linearly increased as the noise level increased. With respect to the

vehicle class, it offered a lower SNR gain, which is 7dB at 20 dB noise level. Thus, among all the four classes, only in the human class do both the algorithms provide improved SNR gains. Similarly, we evaluate the algorithms' performance in the SSNR case, to more rigorously analyze the performance. Figure. 5.2.2 compares the algorithms' performance with four different audio classes in the SSNR metric.

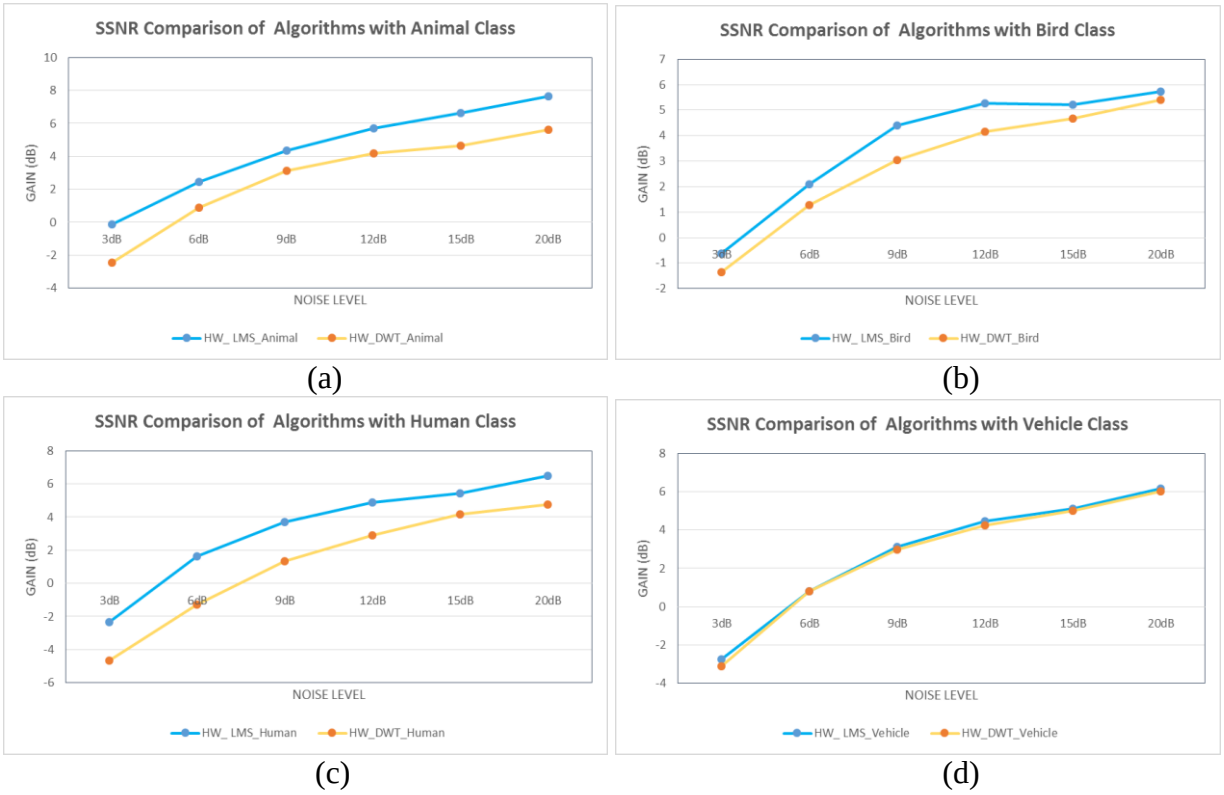


Figure 5.2.2 Hardware SSNR comparison within denoising algorithms with four audio classes: (a) Animal, (b) Bird, (c) Human and (d) Vehicle

From figure. 5.2.2 we see that the LMS algorithm performance is better compared to the DWT algorithm. Most importantly, notice that the LMS only offers improved SSNR performance over the DWT algorithm in animal, bird, and human classes, but not in the vehicle class. However, in the vehicle class both algorithms perform almost same, as depicted in Figure. 5.2.2 (d). In summary, better results can be attained in the animal class compared to the bird, human, and vehicle classes.

Figure. 5.2.3 represents the hardware implementation of the denoising algorithms' performance with respect to the AWGN, PM, and DJI drone noise for the SNR metric.

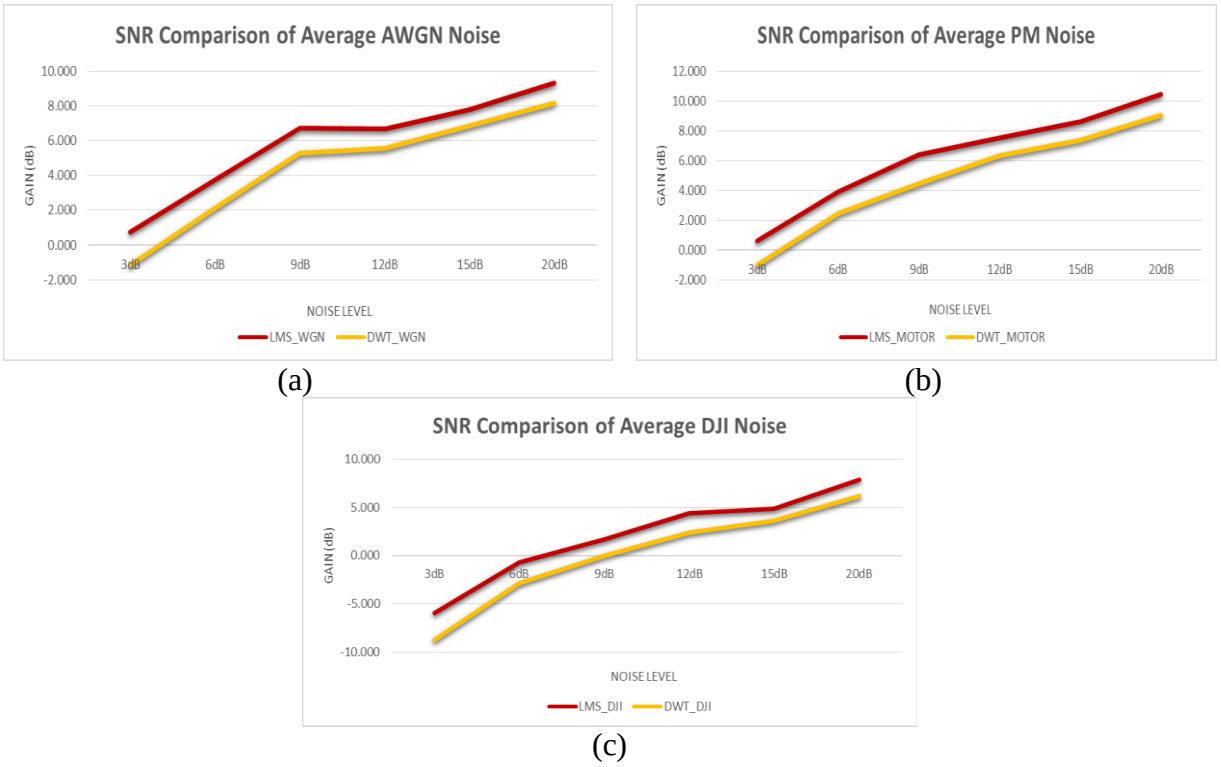


Figure 5.2.3 Hardware noise comparison within denoising algorithms with SNR scenario:
(a) AWGN Noise, (b) Propeller and Motor (PM) Noise, (c) DJI Drone Noise

Figure. 5.2.4 presents the hardware implementation of the denoising algorithms' performance with respect to the AWGN, PM, and DJI drone noise in the SSNR metric. In AWGN and PM noise scenarios, both the LMS and DWT algorithms' SSNR gains increased linearly up to 9 dB, and then were constant. However, in the DJI drone noise scenario, we observed that both algorithms offered negative SSNR, 3 dB at 9 dB noise level, and after that the SSNR gain became positive and started increasing linearly.

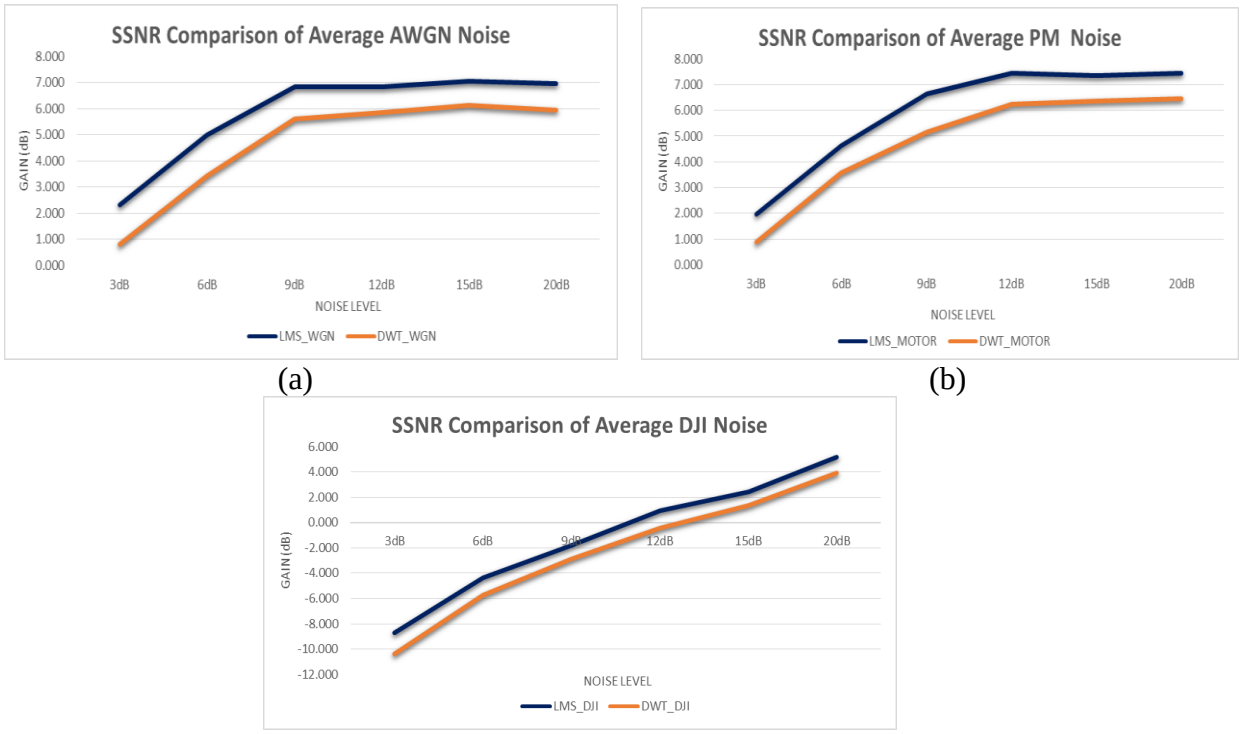


Figure 5.2.4 Hardware noise comparison within denoising algorithms with SSNR scenario: (a) AWGN Noise, (b) Propeller and Motor (PM) Noise, (c) DJI Drone Noise

From the hardware results, we conclude that the DJI drone noise source contains much stronger harmonics compared to AWGN and PM noise scenarios. Figures. 5.2.5 and 5.2.6 show the overall performance of our implemented denoising algorithm with respect to SNR and SSNR metrics.

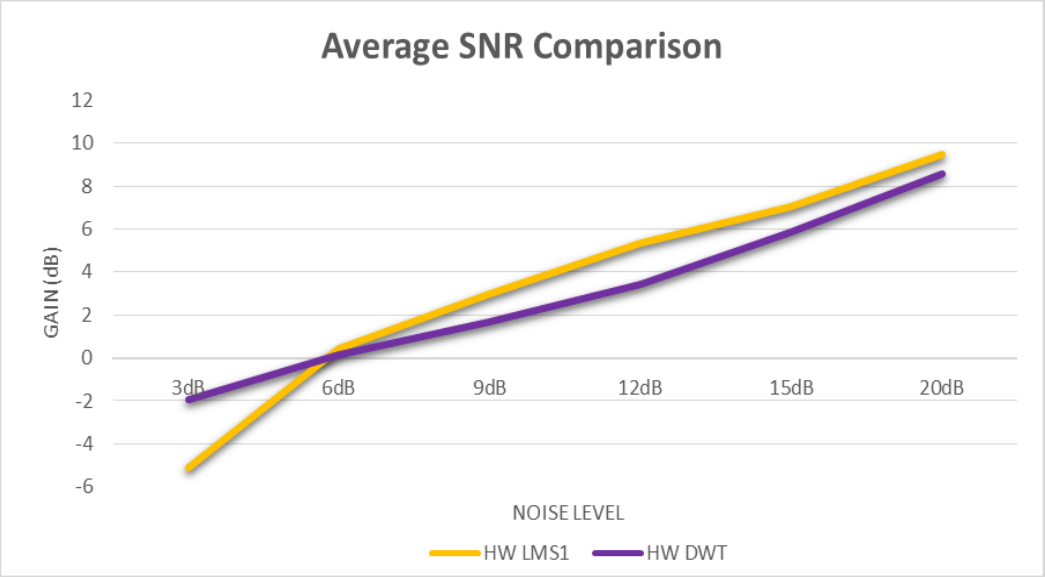


Figure 5.2.5 Hardware overall SNR comparison within denoising algorithms

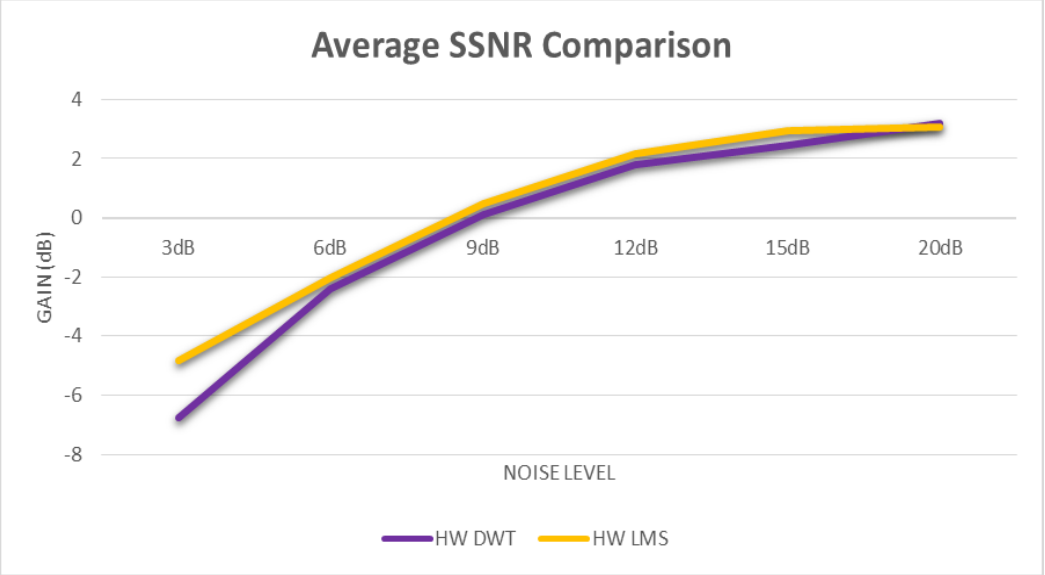


Figure 5.2.6 Hardware overall SSNR comparison within denoising algorithms

Figures. 5.2.5 and 5.2.6, it is clear that the LMS algorithm noise cancellation technique is better than the DWT algorithm. From figure. 5.2.5, we notice that both algorithms obtained negative SNR gain up to 12 dB noise level. But after 12 dB of noise level, both algorithms started offering better SNR gains. Similarly, from figure. 5.2.6, we observe that both algorithms obtain negative SSNR values up

to 9 dB level of noise. But after 9 dB of noise level, both algorithms started offering better SSNR gains. Figures. 5.2.7. and 5.2.8 explain the overall performance of our implemented denoising algorithms with respect to the LLR and LSPD metrics. LMS shows similar performance compare to DWT in both LLR and LSPD metrics.

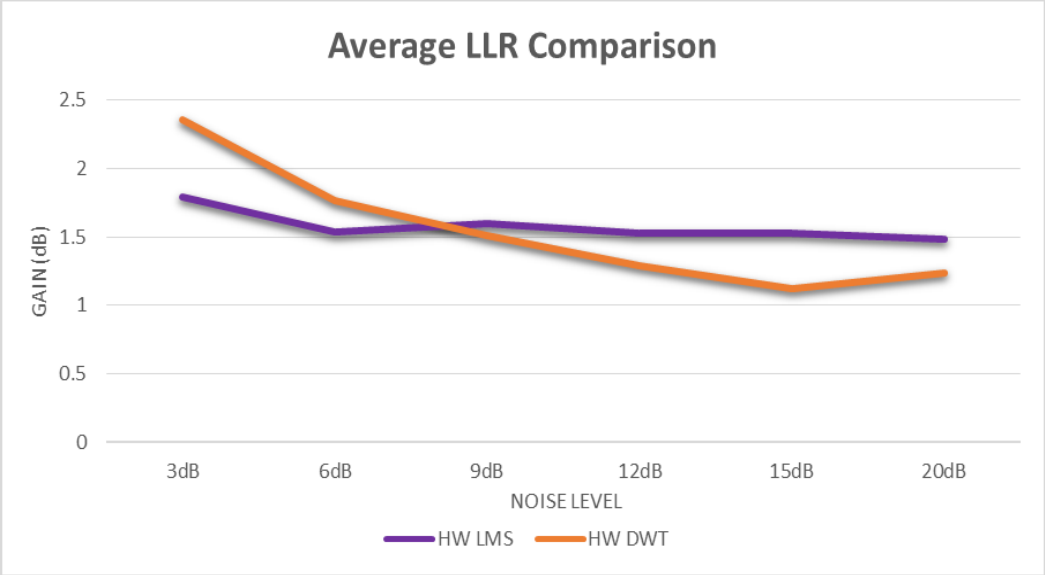


Figure 5.2.7 Hardware overall LLR comparison within denoising algorithms

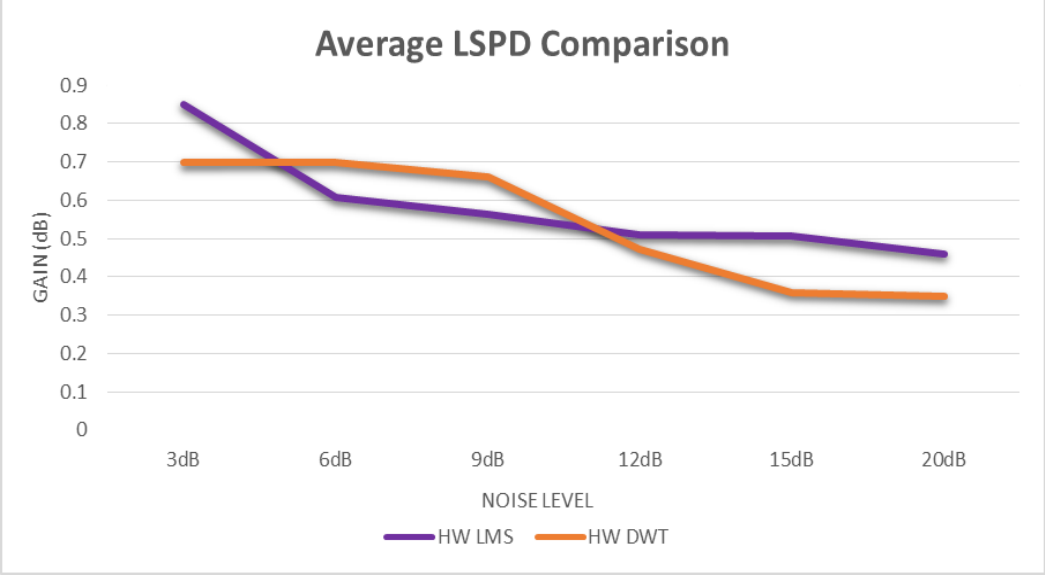


Figure 5.2.8 Hardware overall LSPD comparison within denoising algorithms

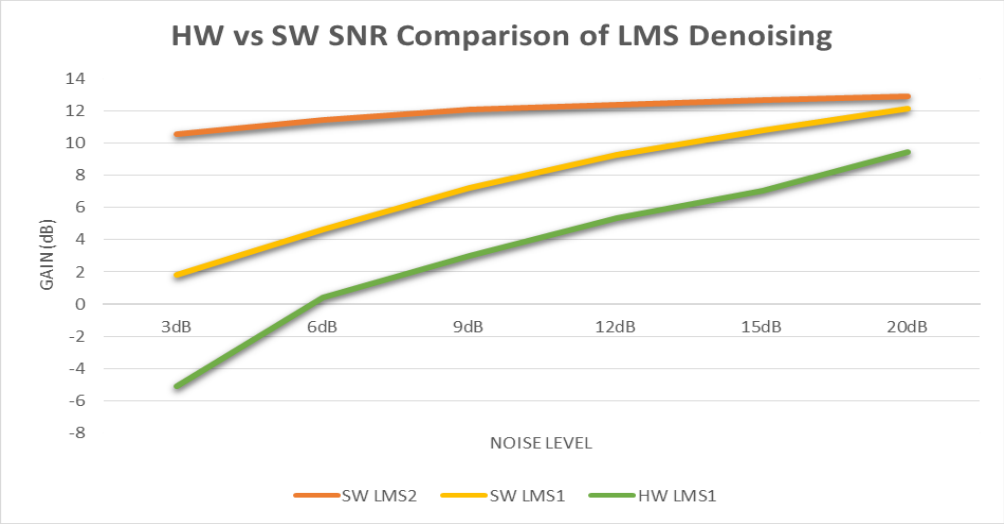


Figure 5.2.9 Hardware vs. software SNR comparison within denoising algorithms

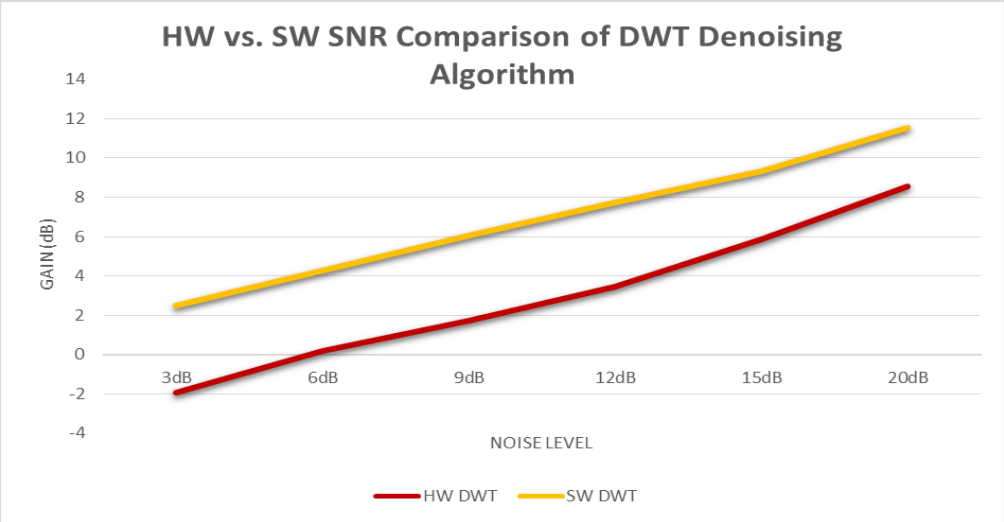


Figure 5.2.10 Hardware vs. software SSNR comparison within denoising algorithms

From figures. 5.2.9 and it is obvious that, software implementation of LMS-2 algorithm performance is better than to hardware LMS-1 and software LMS-1 denosing algorithm with respect to SNR. From Figures. 5.2.10. It is clear that hardware DWT algorithm performance is slightly less than software DWT. However, from above comparison it's clear that LMS is an efficient and better denoising algorithm compare to DWT.

5.3 Classification Results

We now present the results of the software implementation classification experiments conducted to demonstrate the performances of the implemented SVM and NB classifiers. The database is partitioned into a training set and a testing set. For evaluation, 10-fold cross validation is performed. The accuracy of classification is measured by rates of correctly classified samples for testing samples. Figure. 5.3.1 shows the performance of the SVM and NB classifiers with respect to the AWGN noise case, before and after denoising.

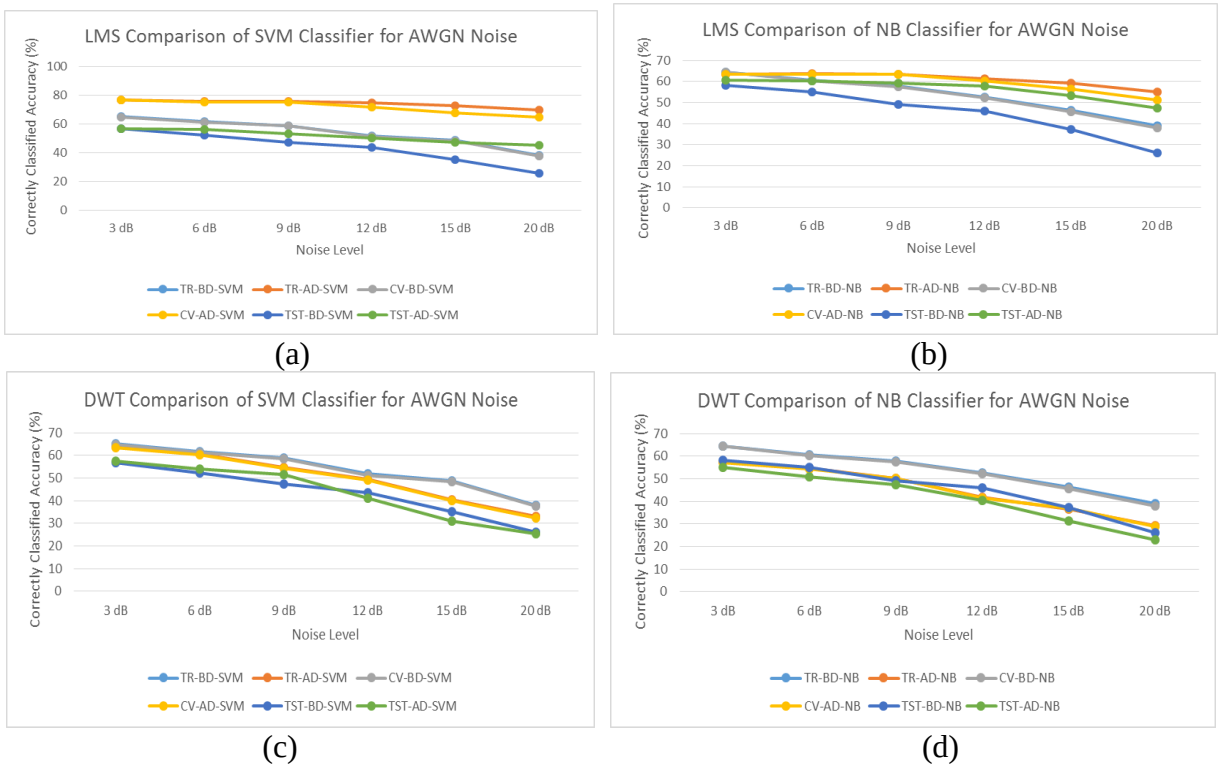


Figure 5.3.1 SVM vs NB classifier performance analysis with respect to AWGN noise in before and after denoising:
(a) LMS with SVM, (b) LMS with NB, (c) DWT with SVM, (d) DWT with NB.

From figure. 5.3.1, we can conclude that the SVM classifier performance is superior compared to the NB classifier, in both the DWT and LMS based denoising conditions. Since the LMS denoising algorithm offered a better denoised signal, meaning a much cleaner signal, this

significantly helped for classification. From figure. 5.3.1 (a), we notice that in the training set, for the SVM classifier, the accuracy reached almost 80%, and that is also perfectly validated by cross validation results. From figure. 5.3.1 (b), we summarize that in the training set, for the NB based classifier, the accuracy reached at almost 68%, and that is also perfectly validated by cross validation results. However, compared to the SVM classifier, the results were not as good. Similarly, from Figure. 5.3.1 (c), for the DWT algorithm, the SVM classifier provided almost 69% accuracy, and that is validated by cross validation results. Thus, we can conclude that LMS based denoised audio signal offered better accuracy for both SVM and NB classifiers compare to DWT based denoised signals. Figure. 5.3.2 illustrates the SVM vs NB classifier performance with respect to UAV's own propeller and motor noise.

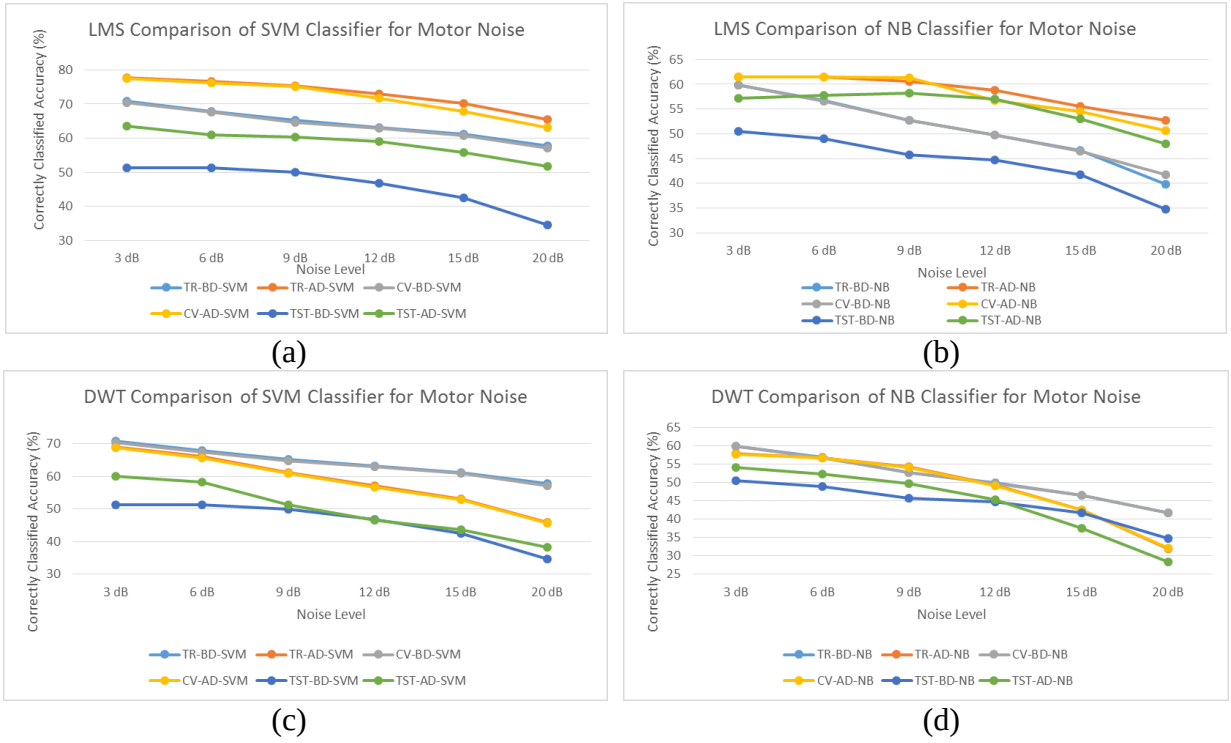


Figure 5.3.2 SVM vs NB classifier performance analysis with respect to PM noise:
(a) LMS with SVM, (b) LMS with NB, (c) DWT with SVM, (d) DWT with NB.

From figure. 5.3.2 we can see that the SVM classifier performance is superior compared to the NB classifier both the DWT and LMS algorithms. Since the LMS denoising algorithm offered a better denoised signal, meaning a much cleaner signal, this significantly improves classification accuracy. From figure. 5.3.2 (a), we notice that in the training set, for the SVM classifier, after denoising, the accuracy reached at almost 78%, and that is also completely validated by cross-validation results. But the performance of classification accuracy decreases as the noise level increases. Before denoising, the classification accuracy reached 71%. The LMS algorithm is a better denoising algorithm compared to the DWT algorithm. From Figure. 5.3.2 (b), we summarize that in the training set, for the NB based classifier, after denoising, the accuracy reached almost 62%, and that is also perfectly validated by cross validation. But before denoising, it offered 59% accuracy. In this, the SVM classifier is better than the NB classifier. Similarly, from Figure. 5.3.2 (c), the SVM classifier after the DWT based denoising provided almost 70 % accuracy, and that is validated by cross-validation results. For Figure 5.3.2 (d) the NB classifier offered 62% accuracy. Thus, we can determine that LMS based denoised audio signals offer better accuracy for both SVM and NB classifiers compare to DWT based denoised signals. Hence, with the above analysis we reach a conclusion that, SVM is an efficient classifier compared to NB, and that the LMS algorithm is better as a denoising algorithm compared to the DWT algorithm. Moreover, complete WEKA simulation results data and their corresponding graphical representations is provided in Appendix-B and Appendix-C consequently.

CHAPTER VI: CONCLUSION

In this work, we implement and evaluate four different denoising algorithm performances. Simulation result shows that the LMS adaptive denoisng and DWT algorithm performance are superior compare to BTH and WF denoising algorithms. We have also implement the LMS and DWT denoising algorithms in C5535 DSP board for hardware test and evaluated their performances. In addition, we have evaluated and analyzed the performance of these algorithms for varying noise power levels and three different noise sources related to UAV operation. Finally, we have implemented and evaluated two classifiers, namely, support vector machines and naive bayes to evaluate the performance of denoising algorithms in target classification application. The SVM classifier outperforms the Naïve Bayes classifier on denoised target audio sources. It is also observed that LMS based denoising algorithm results offered better accuracy compare to DWT based denoised signal during classification.

Appendix A. List of Abbreviations and Acronyms

Unmanned Aerial Vehicles (UAVs)

Signal to Noise Ratio (SNR)

Segmental SNR (Seg-SNR)

Least Mean Square (LMS)

Discrete Wavelet Transform (DWT)

Continuous Wavelet Transform (CWT)

Wiener Filter (WF)

Iterative Wiener Filtering (IWF)

Block Thresholding (BTH)

Additive White Gaussian Noise (AWGN)

Log Likelihood Ratio (LLR)

Log Spectral Distance (LSD)

Short Time Fourier Transform (STFT)

Stein Unbiased Risk Estimator (SURE)

Digital Signal Processing (DSP)

Linear Predictive Coding (LPC)

Mel Frequency Cepstral Coefficient (MFCC)

Support Vector Machine (SVM)

Naïve Bayes (NB)

Code Composer Studio (CCS)

Appendix B. Classification Results

TABLE.1 DWT classification results of propeller and motor noise scenario

TRAINING				
SVM-DWT-PM			**Naive Bayes-DWT-PM**	
Before Denoising (Noisy Data)				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	70.758	29.242	59.8537	40.1463
6	67.9455	32.0545	56.7952	43.2048
9	65.1995	34.8005	52.7327	47.2673
12	63.1981	36.8019	49.8072	50.1928
15	61.1769	38.8231	46.5891	53.4109
20	57.7593	42.2407	41.7952	58.2048
After Denoising				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	69.0027	30.9973	57.7793	42.2207
6	66.004	33.996	56.762	43.238
9	61.0705	38.9295	54.2354	45.7646
12	57.2141	42.7859	49.2952	50.7048
15	53.0253	46.9747	42.5931	57.4069
20	45.9574	54.0426	32.0013	67.9987
CROSS-VALIDATION				
SVM-DWT-PM			**Naïve Bayes-DWT-PM**	
Before Denoising (Noisy Data)				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	70.359	29.641	59.8338	40.1662
6	67.5399	32.4601	56.6356	43.3644
9	64.7074	35.2926	52.6729	47.3271
12	62.9654	37.0346	49.7207	50.2793
15	60.8577	39.1423	46.5359	53.4641
20	57.0745	42.9255	41.8152	58.1848
After Denoising				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified

3	68.7566	31.2434	57.7527	42.2473
6	65.7314	34.2686	56.5891	43.4109
9	60.8511	39.1489	54.1822	45.8178
12	56.7487	43.2513	49.0559	50.9441
15	52.7859	47.2141	42.4202	57.5798
20	45.7048	54.2952	32.1941	67.8059
TESTING				
SVM-DWT-PM			**Naïve Bayes-DWT-PM**	
Before Denoising (Noisy Data)				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	51.3032	48.6968	50.4787	49.5313
6	51.25	48.75	48.9628	51.0372
9	49.9734	50.0266	45.7713	54.2287
12	46.7287	53.2713	44.7074	55.2926
15	42.4734	57.5266	41.8085	58.1915
20	34.6011	65.3989	34.7606	65.2394
After Denoising				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	60.0266	39.9734	54.1489	45.8511
6	58.1383	41.8617	52.3404	47.6596
9	51.1968	48.8032	49.7074	50.2926
12	46.6489	53.3511	45.2926	54.7074
15	43.5904	56.4096	37.4734	62.5266
20	38.1649	61.8351	28.3245	71.6755

TABLE.2 DWT classification results of AWGN noise scenario.

TRAINING				
SVM-DWT-AWGN			**NaiveBayes-DWT-AWGN**	
Before Denoising (Noisy Data)				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	65.2327	34.7673	64.5213	35.4787
6	61.6955	38.3045	60.6316	39.3684
9	58.8564	41.1436	57.8059	42.1941
12	51.8218	48.1782	52.7926	47.2074
15	48.8231	51.1769	46.2766	53.7234
20	38.1383	61.8617	38.8896	61.1104
After Denoising				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	63.9561	36.0439	57.3737	42.6263
6	60.6981	39.3019	54.5678	45.4322
9	54.887	45.113	50.1729	49.8271
12	49.5213	50.4787	41.7354	58.2646
15	40.4521	59.5479	36.7553	63.2447
20	32.9322	67.0678	29.1556	70.8444
CROSS-VALIDATION				
SVM/SMO-DWT-AWGN			**NaiveBayes-DWT-AWGN**	
Before Denoising (Noisy Data)				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	64.9535	35.0465	64.5213	35.4787
6	61.4761	38.5239	60.3923	39.6077
9	58.5572	41.4428	57.6463	42.3537
12	51.4096	48.5904	52.3604	47.6396
15	48.4907	51.5093	45.6582	54.3418
20	37.6729	62.3271	37.9521	62.0479
After Denoising				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	63.4973	36.5027	57.3005	42.6995
6	60.3258	39.6742	54.4415	45.5585
9	54.4681	45.5319	50.106	49.8936

12	49.0359	50.9641	41.5891	58.4109
15	40.0532	59.9468	36.7886	63.2114
20	32.3404	67.6596	28.9761	71.0239
TESTING				
SVM-DWT-AWGN			**NaiveBayes-DWT-AWGN**	
Before Denoising (Noisy Data)				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	56.9681	43.0319	58.1383	41.8617
6	52.1543	47.8457	55.1862	44.8138
9	47.3936	52.6064	49.2287	50.7713
12	43.5904	56.4096	46.1436	53.8564
15	35.3457	64.6543	37.4468	62.5532
20	25.984	74.016	26.0638	73.9362
After Denoising				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	57.4202	42.5798	55.2394	44.7606
6	54.0957	45.9043	50.7979	49.2021
9	51.6755	48.3245	47.2606	52.7394
12	41.0904	58.9096	40.3989	59.6011
15	31.1702	68.8298	31.3564	68.6436
20	25.3989	74.6011	23.0585	76.9415

TABLE.3 LMS classification results of propeller and motor noise scenario.

TRAINING				
SVM-LMS-PM			**NaiveBayes-LMS-PM**	
Before Denoising (Noisy Data)				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	70.758	29.242	59.8537	40.1463
6	67.9455	32.0545	56.7952	43.2048
9	65.1995	34.8005	52.7327	47.2673
12	63.1981	36.8019	49.8072	50.1928
15	61.1769	38.8231	46.5891	53.4109
20	57.7593	42.2407	41.7952	58.2048
After Denoising				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	77.6862	22.3138	61.4628	38.5372
6	76.5691	23.4309	61.516	38.484
9	75.4122	24.5878	61.609	38.391
12	75.0399	24.9601	61.8351	38.1649
15	75.1995	24.8005	61.6157	38.3843
20	74.4827	25.5173	60.7327	39.2673
CROSS-VALIDATION				
SVM-LMS-PM			**Naïve Bayes-LMS-PM**	
Before Denoising (Noisy Data)				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	70.359	29.641	59.8338	40.1662
6	67.5399	32.4601	56.6356	43.3644
9	64.7074	35.2926	52.6729	47.3271
12	62.9654	37.0346	49.7207	50.2793
15	60.8577	39.1423	46.5359	53.4641
20	57.0745	42.9255	41.8152	58.1848
After Denoising				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	77.4269	22.5731	61.5226	38.4774
6	76.3098	23.6902	61.4495	38.5505
9	75.0731	24.9269	61.4029	38.5971

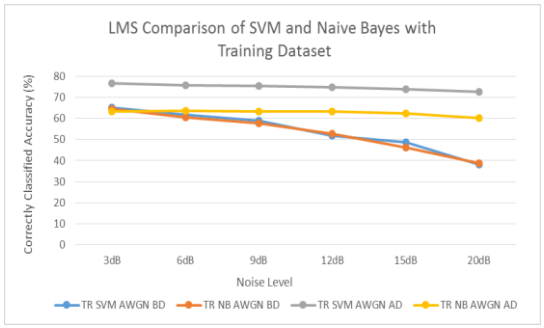
12	74.7074	25.2926	61.7686	38.2314
15	74.7673	25.2327	61.4894	38.5106
20	76.0838	23.9162	62.5931	37.4069
TESTING				
SVM-LMS-PM			**Naïve Bayes-LMS-PM**	
Before Denoising (Noisy Data)				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	51.3032	48.6968	50.4787	49.5313
6	51.25	48.75	48.9628	51.0372
9	49.9734	50.0266	45.7713	54.2287
12	46.7287	53.2713	44.7074	55.2926
15	42.4734	57.5266	41.8085	58.1915
20	34.6011	65.3989	34.7606	65.2394
After Denoising				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	63.5904	36.4096	57.1543	42.8457
6	61.0106	38.9894	57.8191	42.1809
9	60.3457	39.6543	58.2713	41.7287
12	59.0957	40.9043	58.0319	41.9681
15	60.7181	39.2819	59.0691	40.9309
20	56.7287	43.2713	58.0053	41.9947

TABLE.4 LMS classification results of AWGN noise scenario.

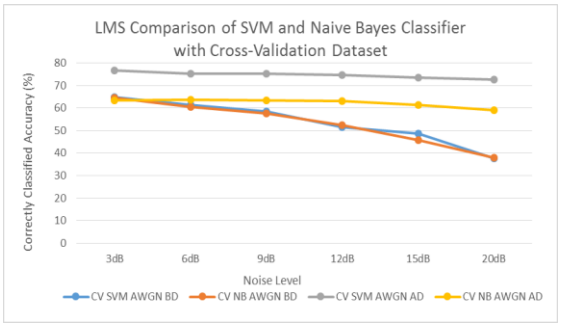
TRAINING				
SVM-LMS-AWGN			**Naïve Bayes-LMS-AWGN**	
Before Denoising (Noisy Data)				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	65.2327	34.7673	64.5213	35.4787
6	61.6955	38.3045	60.6316	39.3684
9	58.8564	41.1436	57.8059	42.1941
12	51.8218	48.1782	52.7926	47.2074
15	48.8231	51.1769	46.2766	53.7234
20	38.1383	61.8617	38.8896	61.1104
After Denoising				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	79.8816	20.1184	67.4774	32.5226
6	75.7713	24.2287	63.7766	36.2234
9	75.6184	24.3816	63.4441	36.5559
12	74.9069	25.0931	63.3112	36.6888
15	73.9362	26.0638	62.4468	37.5532
20	72.8324	27.1676	62.1915	37.8085
CROSS-VALIDATION				
SVM/SMO-LMS-AWGN			**Naïve Bayes- LMS-AWGN**	
Before Denoising (Noisy Data)				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	64.9535	35.0465	64.5213	35.4787
6	61.4761	38.5239	60.3923	39.6077
9	58.5572	41.4428	57.6463	42.3537
12	51.4096	48.5904	52.3604	47.6396
15	48.4907	51.5093	45.6582	54.3418
20	37.6729	62.3271	37.9521	62.0479
After Denoising				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	79.6822	20.3178	63.4043	36.5957
6	75.3191	24.6809	63.5971	36.4029
9	75.2194	24.7806	63.3245	36.6755

12	74.5944	25.4056	63.1715	36.8285
15	73.5838	26.4162	62.4202	37.5798
20	72.5332	27.4668	63.0718	36.9282
TESTING				
SVM-LMS-AWGN			**Naïve Bayes-LMS-AWGN**	
Before Denoising (Noisy Data)				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	56.9681	43.0319	58.1383	41.8617
6	52.1543	47.8457	55.1862	44.8138
9	47.3936	52.6064	49.2287	50.7713
12	43.5904	56.4096	46.1436	53.8564
15	35.3457	64.6543	37.4468	62.5532
20	25.984	74.016	26.0638	73.9362
After Denoising				
dB	Correctly Classified	Incorrectly Classified	Correctly Classified	Incorrectly Classified
3	56.8351	43.1649	60.5053	39.4947
6	56.2766	43.7234	60.3191	39.6809
9	59.3351	40.6649	59.3617	40.6383
12	58.2447	41.7553	59.867	40.133
15	60.4521	39.5479	58.3777	41.6223
20	58.2713	41.7287	56.5426	43.4574

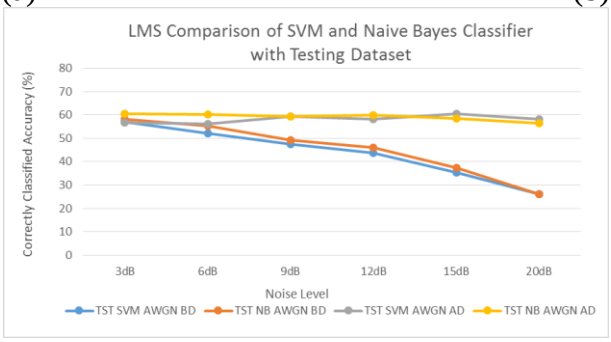
Appendix C. Graphical Representation of Classification Results



(a)

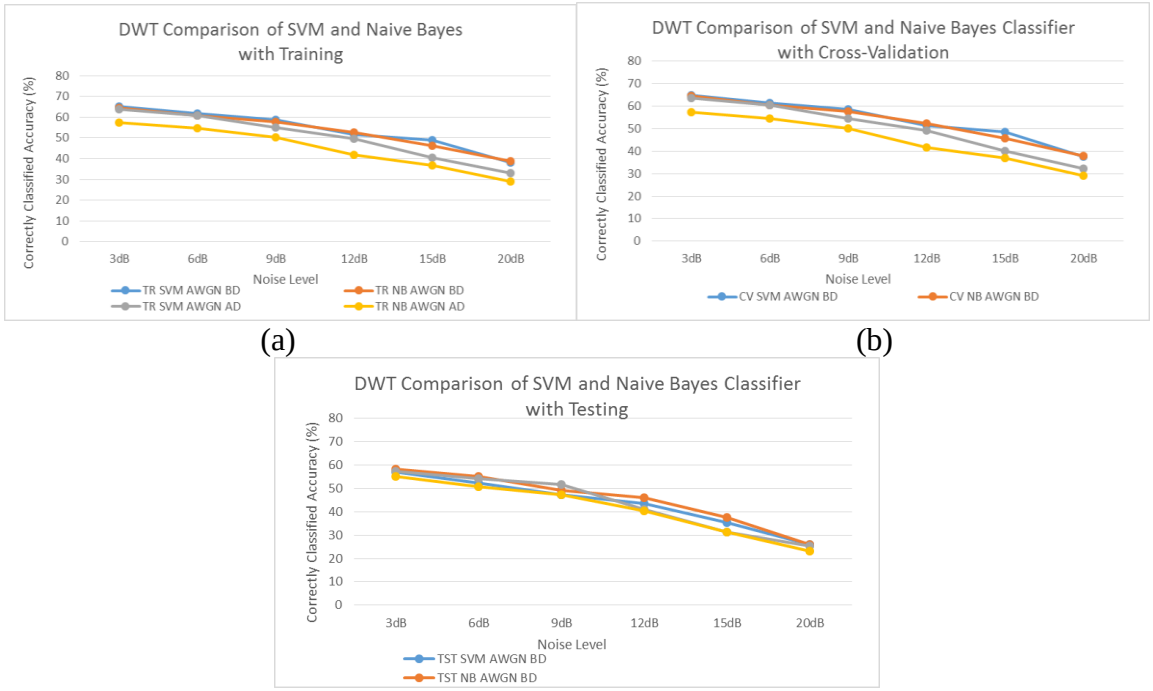


(b)

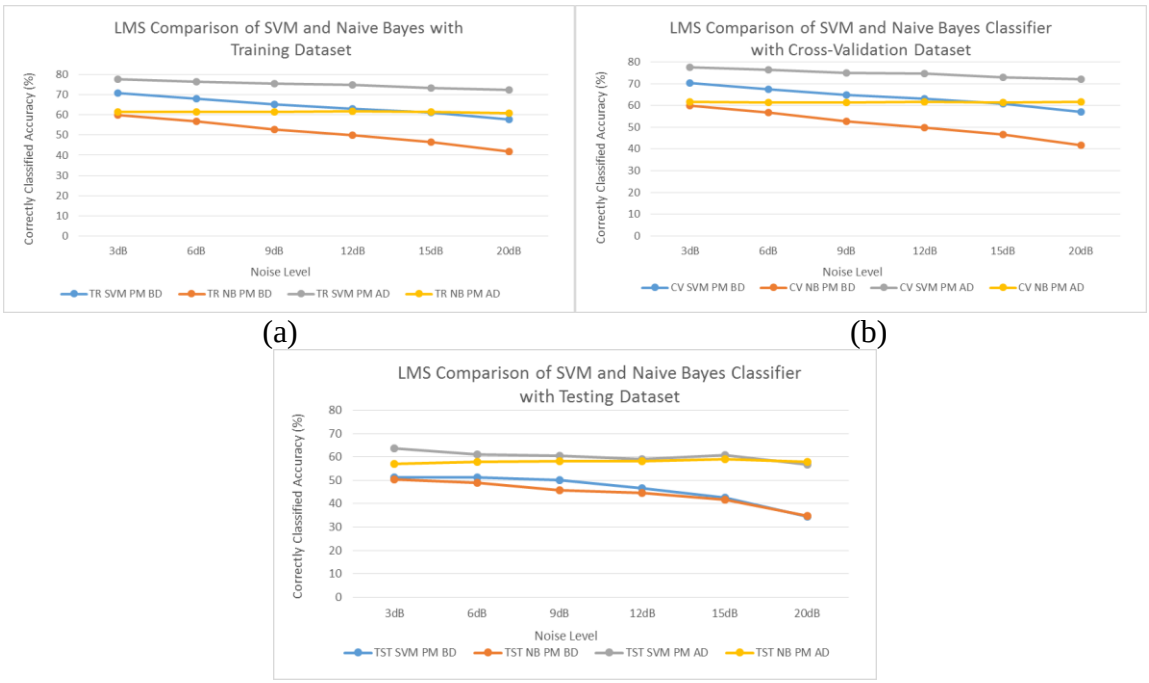


(c)

C .1- SVM vs NB classifier performance analysis with respect to before and after denoising with LMS algorithm: (a) Training data set with SVM vs. NB, (b) Cross-validation data set with SVM vs. NB (c) Testing data set with SVM vs. NB.



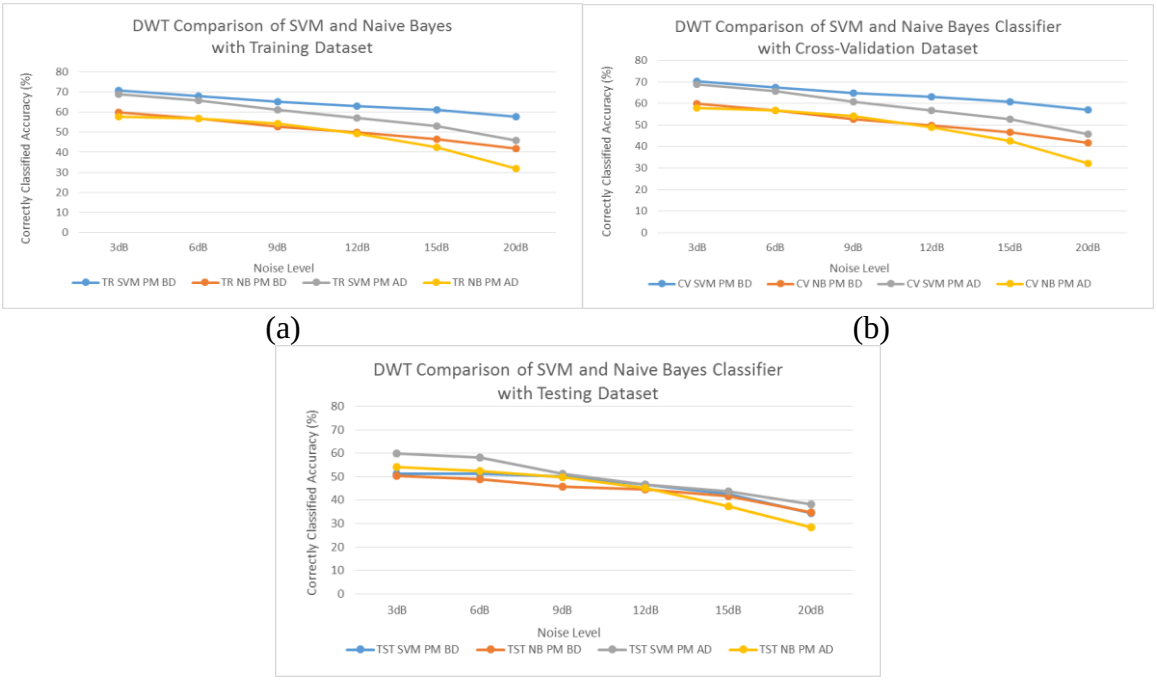
C .2- SVM vs NB classifier performance analysis with respect to before and after denoising with DWT: (a) Training data set with SVM vs. NB, (b) Cross-validation data set with SVM vs. NB (c) Testing data set with SVM vs. NB.



(a) (b)

(c)

C.3- SVM vs NB classifier performance analysis with respect to PM noise in before and after denoising with LMS: (a) Training data set with SVM vs. NB, (b) Cross-validation data set with SVM vs. NB (c) Testing data set with SVM vs. NB.



C.4- SVM vs NB classifier performance analysis with respect to PM noise in before and after denoising with LMS: (a) Training data set with SVM vs. NB, (b) Cross-validation data set with SVM vs. NB (c) Testing data set with SVM vs. NB.

REFERENCES

[1] P. Marmaroli and X. Falourd *et.all*, "A UAV motor denoising technique to improve localization of surrounding noisy aircrafts: proof of concept for anti-collision systems", *IEEE Acoustics, Speech and Signals*, pp.333-351, 2012.

[2] Y. Wu and W. Zhao, "The Improvement of audio noise reduction system based on LMS algorithm", *International Conference on Computer Science and Application*, pp. 590-594, 2013.

[3] C. Gargour, M. Gabrea, V. Ramachandran and J. Lina. "A Short Introduction to Wavelets and Their application", *IEEE Circuits and System Magazine*, ISSN: 1531-636X, vol. 2, pp. 57-67, 2009.

[4] T. Cai, "Adaptieve wavelet estimation: A block thresholding and oracle inequality approach," *Ann Statist*., vol.27, pp. 898-924, 1999.

[5] J. A. Reyes , *et. all.*, "DSP-based oversampling adaptive noise canceller for background noise reduction for mobile phones," *IEEE CONIELECOMP*, pp. 327-332, 2012.

[6] L. Xu *et. all*, "Research on LMS adaptieve filtering algorithm for acoustic emission signal processing" *In proceeding of the 8th World Congress on Intelligent Control and Automation*, pp. 7037-7040, July 6-9, 2010.

[7] N. J. Bershad, "Analysis of the normalized LMS algorithm with Gaussian inputs", *IEEE Trans. Acoust., Speech, Signal Process*., vol.34, no. 4, pp. 793–806, Aug. 1986.

[8] HX. Zheng, H. Liu, "Noise Cancellation Based on Adaptive Filter", *Information and Electronic Engineering*, vol.5, no.6, pp.444-448, 200.

[9] Daniel J. Allred, Venlatesh Krishnan, "LMS Adaptive Filters Using Distributed Arithmetic for High Throughput Time-Sequenced Adaptive Filter*", IEEE Transaction on Circuits and Systems,* vol.52, no.7, pp.1327-1337, 2005.

[10] DP Das, G Panda, Active Mitigation of Nonlinear Noise Processes Using a Novel Filtered-s LMS Algorithm", *IEEE Transactions on Speech and Audio Processing*, pp. 313-322, May, 2004.

[11] SM Kuo, DR Morgan, "Active noise control: a tutorial review", *In proceedings of the IEEE*, 87(6):943-973, June, 1999.

[12] S. C. Douglas, T. H. Y. Meng, "Normalized data nonlinearities for LMS adaptation", *IEEE Trans. Signal Process*., pp. 1352–1354, Jun.1994.

[13] LIU Jian-feng, "A novel adaptation scheme in the NLMS algorithm for echo cancellation", *IEEE Signal Processing Letters*, 8(1):20-22, 2001.

[14] S. G. Mihov, R. M. Ivanov, A. N. Popov, "Denoising speech signals by wavelet transform" *Annual Journal of Electronics*, ISSN: 1313-1842, 2009.

[15] A. Grossmann, and J. Morlet, "Decomposition of Hardy functions into square integrable wavelets of constant shape", *SIAM Journal of Analysis, 15:* 723-736, 1984.

[16] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis", *IEEE Transformation and Information Theory* 36: 961- 1005, 1990.

[17] L. Debnath, "Wavelet Transformation and their Applications", Birkhäuser Boston, 2002.

[18] D. Donoho, "De-Noising by Soft-Thresholding", *IEEE Transaction on Information Theory*, pp. 613- 627, 1995.

[19] C. Taswell. "The What, How and Why of Wavelet Shrinkage Denoising", *IEEE Computing in Science and Engineering*, ISSN: 1521-9615, vol. 2, no. 3, pp. 12-19, June 2009.

[20] R. C. Nongpiur, D. J. Shpak, "Impulse-noise suppression in speech using the stationary wavelet transform", *Acoustical Society of America*, pp. 866-879, 2013.

[21] M. Saifuzzaman, *et. All*, "Application of wavelet transform and its advantages compared to fourier transform", *Journal of Physical Science*, ISSN:1521-9615, vol. 13, pp. 121-134, 2009.

[22] J. Rox and E. Vincent. "Consistent wiener filtering for audio source separation", *IEEE Signal Processong Letters*, vol. 20, no. 3, pp. 217-220, 2013.

[23] L. Ke, W. yuan, Y. Xiao, "An Improved Wiener Filtering Method in wavelet Domain," *IEEE Audio, Language and Image Processing Conference Ann. Statist*, pp. 1527-1531, 2008.

[24] R. Mao, Y. Zhou and H. Liu, "An improved iterative wiener filtering algorithm for speech enhancement", *IEEE International Conference on Computers and Communications (ICCC)*, pp. 436-440, 2015.

[25] Y. Guoshen, and S. Mallat, and E. Bacry, "Audio Denoising by Time-Frequency Block Thresholding," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, pp. 1830-1839, 2008.

[26] V. Harini, *et. all*, "Block hresholding algorithm for enhancement of an audio signal corrupted by noise", *IJRRAS*, vol. 7, no. 5, pp. 213-217, May, 2011.

[27] T. Cai and B. W. Silverman, "Incorporation information on neighbouring coefficiecnts into wavelet estimation," *Sankhya*., vol. 63, pp. 127-148, 2001.

[28] T. Cai and H. Zhouo, "A data driven block thresholding approaches to wavelet estimation," *Statistics Dept., Univ. of Pennsylvania*, Tec. Rep., 2005.

[29] R. Martin, "Noise power spectral density estimation based on optimal smoothing and minimum statistics", *IEEE Trans. Speech Audio Process*., vol. 9, no. 5, pp. 504-512, 2001.

[30] S. Chokkalingam, and K. Komathy, "Comparison of different classifier in WEKA for rheumatoid arthritis", *IEEE International Conference on Human Computer Interactions (ICHCI)*, pp. 1–6, 2013.

[31] Z. Liu, Y. Wang, and T. Chen, "Audio feature extraction and analysis for scene segmentation and classification", *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, vol. 20, no. 1-2, pp. 61–79, 1998.

[32] B. Liang, *et. all*, "Feature Analysis and Extraction for Audio Automatic Classification", *in Proc. ICASSP*, vol. 1, pp. 767–772, 2005.

[33] J. T. Geiger, *et. all*, " Large scale audio feature extraction and SVM for acoustic scene classification", *IEEE Workshop on Application of Signal Processing to Audio and Features*, pp. 1–4, 2013.

[34] H. Jiang, *et. all*., "SVM-based audio scene classification", *IEEE in Proc. Natural Language Processing and Knowledge Engineering (NLP-KE)*, pp. 131–136, 2005.

[35] J. Vavrek, A. Cizmar, and J. Juhar, "SVM binary decision tree architecture for multi-class audio classification," *IEEE 54th Annual Symposium ELMAR*, pp. 202–206, 2012.

[36] Zhouyu. Fu, *et. all*, "Learning naive bayes classifiers for music classification and retrieval," *IEEE International Conference on Pattern Recognition*, pp. 4589–4592, 2010.

[37] A. McCallum and K. Nigam, "A comparison of event models for Naive Bayes text classification*", In AAAI Workshop on Learning for Text Categorization*, 1998.

[38] J. Ren, *et. all*, "Naïve bayes classification of uncertain data", *IEEE International Conference on Data Mining*, pp. 944-949, 2009.

[39] M. Ghazanfar and A. Bennett, "An improved switching hybrid recommender system using naïve bayes classifier and collaborative filtering", *IEEE IMECS*, pp. 944-949, 2010.

[40] N. Carlos, *et. all*, "Feature selection in automatic genre classification", *IEEE International Symposium on Multimedia*, pp. 39-44, 2008.

[41] B. Yang, "A study of inverse short-time Fourier transform", *In proc. ICASSP*, Apr.2008, pp. 3541-3544.

[42] C. M. Bishop, *Pattern recognition and machine learning*. New York: Springer, 2006, pp. 85-87.

[43] TMS320C5505/5535 Evaluation Module Users Guide, Texas Instrument.

[44] TMS320C5505/5535 Code Composer studio Tutorial, Texas Instrument.

[45] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

Curriculum Vitae

AHMED SONY KAMAL CHOWDHURY

Office: SEB-4219, Electrical & Computer Engineering Dept., University of Nevada, Las Vegas.

SKILLS

Programming: C language, MATLAB/SIMULINK etc.
Software: MATLAB, CADENCE, P-Spice, LT-Spice, etc.
Hardware: TMS320C5535 DSP Development Kit, Basic Soldering and Rework.
Computer Skills: Experiences with Microsoft Word, Numbers/Excel & Keynote/Power Point, Mac OS, Linux etc.

EDUCATION

Master of Science in Electrical Engineering
University of Nevada Las Vegas, Las Vegas, USA.

Bachelor of Science in Electrical & Electronic Engineering
North South University, Dhaka, Bangladesh.

WORKING EXPERIENCE

Graduate Research Assistant (Fall-14 and Spring-15)
Advisor: Dr. Venkatesan Muthukumar (Associate Professor)
University of Nevada Las Vegas, ECE Department.

Graduate Teaching Assistant (Fall-15 & Spring-16)
Duties including grading quizzes and assignments, teaching the lab class, evaluate student's lab reports, monitor student progress, and prepare reports for supervisor etc.

ACHIEVEMENT & ACTIVITIES

Scholarship: Obtained Satisfactory Academic Performance (SAP) scholarship from UNLV.
Society & Activities: Student member of IEEE.