TÉCNICO LISBOA

Speech Recognition for a Small Aerial Robot

Ana Catarina Monteiro Ribeiro de Sá

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Dr. Meysam Basiri
Prof. Pedro Manuel Urbano de Almeida Lima

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Dr. Meysam Basiri
Member of the Committee: Dr. Carlos Jorge Andrade Mariz Santiago

June 2019

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Declaração

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

Acknowledgments

First of all, I would like to thank Dr. Meysam Basiri for giving me the opportunity to develop this thesis under his supervision. It was very important for me to be able to work in a subject that I've been interested for so long and that is not much studied during the degree. His knowledge, insights and suggestions challenged me to learn and use important technologies that otherwise I would be finishing my degree without knowing much about and that is the best reward that I take from this work. I am very thankful for the motivation that he has given me to contribute to this field of studies.

To my mother, father and sister thank you for always believing in me, pushing me to give my best every time and also picking me up when it was necessary.

To my two best friend, Mafalda and Sofia, thank you for being with me for 21 years. Your unconditional support was fundamental to go through these 9 months. Let's now move on to our next chapter.

I give a big thank you to all my friends from University and Erasmus. These past years were amazing and every single one of you has contributed in some way for my growth as a person and as a student, even if you were only teaching me what I shouldn't do. No one would be finishing this chapter of our lives if it wasn't for the support and help we gave each other. We were all there when passing a course seemed impossible and also when we had to celebrate the success of each and every one of us. I have to give a special thank to Leonor, your help and friendship had a great impact on my path at university. Also, thank you so much for the group of people that took some of their time to help me with my thesis. It was not your problem and you wouldn't gain anything with it, still, you spent some hours helping me when I was desperate.

Lastly, I would like to thank my friends from high school for being so positive about my work and for helping me to have that positive mindset during these months.

Thank you.

Abstract

Unmanned Aerial Vehicles are aircrafts that allow the gathering of data in remote areas of difficult access. Capturing audio signals with a drone triggers a series of new applications such as in the field of Search and Rescue. The reason why Speech Recognition hasn't been widely developed for drones is the corrupting noise the engines add to the speech signals. The main goal of this research is to present a study on how to overcome the bad speech recognition performances due to this specific type of additive noise. The implementation and testing of Wiener Filter, Least-Mean-Square Adaptive Filter, Noise Gate, Motion-Dependent Spectral Subtraction and Discrete Wavelet Transform was done for different levels of Signal-to-Noise Ratio. Due to the very promising results, the Least-Mean-Square Adaptive Filter is chosen for further improving a Speech Recognition System. A Feedforward Neural Network that predicts the filter's coefficients from the information on the velocities of the 4 motors is developed and integrated in the system resulting in a better performance. The last proposal of change in the filter is using as input two signals with different Signal-to-Noise Ratio values instead of the noisy speech and the noise-only speech signals. Lastly, a Recurrent Neural Network for Speech Recognition trained with speech corrupted by the drone's ego-noise was also tested but this approach didn't show to be competitive with the previous implementation. Experimental results show that using the Least-Mean-Square Adaptive Filter with filter's coefficients computed by a Neural Network and using two noisy signals with different Signal-to-Noise Ratio values is a good solution for the problem explored and presents robustness against the high levels of noise present.

Keywords

Speech Recognition; Unmanned Aerial Vehicles; Noisy Speech Signal; Least-Mean-Square Adaptive Filter; Ego-noise Cancellation.

Resumo

Veículos aéreos não tripulados são aeronaves que permitem a recolha de dados em áreas de difícil acesso. Capturar áudio com um drone despoleta uma série de novas aplicações. A razão que leva sistemas de Reconhecimento de Fala a não terem sido muito explorados para drones é devido ao ruído que os motores introduzem nos sinais de fala. O objetivo principal desta investigação é apresentar soluções para superar os maus desempenhos de sistemas de reconhecimento de fala devido a este tipo específico de ruído aditivo. A implementação e teste do Filtro de Wiener, Filtro Adaptativo de Mínimos Quadrados, Noise Gate, Subtracção Espectral dependente de Movimento e Transformada Discreta de Wavelet foi realizada para diferentes valores de relação Sinal-Ruído. Devido aos seus resultados, o Filtro Adaptativo de Mínimos Quadrados é escolhido para integrar o Sistema de Reconhecimento de Fala. Uma Rede Neuronal que prevê os coeficientes do filtro a partir das velocidades dos 4 motores é desenvolvida para o sistema, resultando num melhor desempenho do mesmo. A última proposta de alteração no filtro é utilizar como entrada dois sinais com diferentes valores de relação Sinal-Ruído em vez de utilizar um sinal apenas de ruído e outro de fala com ruído. Por fim, uma Rede Neuronal Recorrente para Reconhecimento de Fala treinada com fala corrompida pelo ruído do drone também foi testada, mas esta abordagem não mostrou ser competitiva com a implementação anterior. Resultados experimentais mostram que utilizar o Filtro Adaptativo de Mínimos Quadrados com os coeficientes do filtro a serem calculados por uma Rede Neuronal e utilizar dois sinais ruidosos com diferentes valores de relação Sinal-Ruído é uma solução promissora para este problema em específico.

Palavras Chave

Reconhecimento de Fala; Veículos Aéreos Não Tripulados; Sinais de Fala com Ruído; Filtro Adaptativo de Mínimos Quadrados; Cancelamento de Ruído Próprio.

Contents

List of Figures

List of Tables

Acronyms

ASR	Automatic Speech Recognition
HRI	Human Robot Interaction
ISR	Institute for Systems and Robotics
MAV	Micro Aerial Vehicles
UAV	Unmanned Aerial Vehicle
SNR	Signal-to-Noise Ratio
FNN	Feedforward Neural Network
RNN	Recurrent Neural Network
WER	Word Error Rate
SR	Speech Recognition
DNN	Deep Neural Networks
MFCC	Mel-Scale Frequency Cepstral Coefficients
PLP	Perceptual Linear Prediction
CMS	Cepstral Mean Subtraction
PDF	Probability Density Function
HMM	Hidden Markov Models
HMM/ANN	Hybrid Hidden Markov Models/ Artificial Neural Networks
NN	Neural Networks
SLM	Stochastic Language Models

GMM Gaussian Mixture Models

DL Deep Learning

MSE Mean Squared Error

CNN Convolutional Neural Network

LSTM Long-Short Term Memory

GSC General Sidelobe Canceller

STFT Short-Time Fourier Transform

RIR Room Impulse Response

NMFD Non-negative Matrix Factor Deconvolution

PS-DNN Partially-Shared Deep Neural Network

iGSVD-MUSIC MUltiple SIgnal Classification based on incremental Generalized Singular Value Decomposition

MMSE Minimum Mean Squared Error

TSRT Two-Step Noise Reduction Technique

HRNR Harmonic Regeneration Noise Reduction

DD Decision Direct Approach

PSD Power Spectral Density

LMS Least Mean Square

dB Decibel

ISTFT Inverse Short-Time Fourier Transform

DFT Discrete Fourier Transform

IDFT Inverse Discrete Fourier Transform

DWT Discrete Wavelet Transform

FT Fourier Transform

CTC Connectionist Temporal Classification

IST	Instituto Superior Técnico
ROS	Robot Operating System
API	Application Programming Interface
MDSS	Motion-Dependent Spectral Subtraction
hl	hidden layers

1

Introduction

Contents

On this chapter we will be doing a brief introduction to the proposed theme of study. Furthermore, the objectives to be achieved through the research will be set and the thesis layout will be explained.

1.1 Motivation

Speech Recognition has been a subject of great interest for some decades. Starting by simple comparisons, moving to Hidden Markov Models and now having a big improvement with the arise of Deep Neural Networks, this technology will enable a series of new applications which lead to the improvement of the quality of life of the Humanity. Although there are already lots of applications such as intelligent assistants and voice control in cars which demand a good Automatic Speech Recognition (ASR) system, these applications are not widely used since there is still lots of difficulties to overcome in order to create robust and almost infallible systems. Nevertheless we have to keep in mind the important role that speech recognition has in improving Human Robot Interaction (HRI). Allowing communication with robots through speech facilitates the inclusion of robots in everyday life, specially in communities that didn't have much access to technology such as elderly people.

On the other hand, the number of small aerial vehicles is exponentially increasing and its' utility has been mostly focused on video applications. The usage of small aerial vehicles enabled the possibility of rapidly reaching places that humans couldn't due to obstacles. This started to be a real advantage when Computer Vision was associated to small aerial vehicles. A big contribution given by this merging was the fact that it allowed rescue teams to be able to look further for people in a catastrophe situation. Having a camera in a drone also led to an increase of captured data that can be used in a series of areas of business allowing their development and growth.

Since the two previous technologies mentioned brought a great development to science and business, we want to focus on the possibilities of joining the two. With this project we want to study what are the applications possible, the adversities of implementation and the best methods to implement speech recognition and voice control in small aerial robots.

The applications for this type of technology are endless. Life jackets for airplanes or boats usually have a whistle. We can design the drones to recognize the sound of those whistles in case of the occurrence of an accident. With a drone it would be much easier to access areas such as the open sea or places which sometimes, because of the disaster, are of very difficult access and detect if there are survivors and what is their location. The drone would decrease the response time and would increase the probability of saving more people.

In indoor environments we can also take advantage of the usage of a drone. When in a conference in a big auditorium, it gets difficult for the audience to ask questions to the speaker since someone has to go across the room and handover a microphone. We can implement a system in a small drone for it to

go closer to the person that wants to ask a question and use its own microphone to amplify what the person is saying in order for everyone in the room to listen.
The idea of integrating speech recognition with drones came due to the development of a project in Institute for Systems and Robotics (ISR). This bigger project consists in a robot that works in a domestic environment which has to interact with people inside the house and help those people performing tasks. Using a small aerial vehicle can help in energy saving since instead of having the bigger robot going around the house, we can send the drone which spends less energy to move, has a reduced response time because it will be faster to reach certain points and can maintain easy interaction with the user since the user can speak to the drone and the latter can transmit the information to the bigger robot.
Conjugating pre-processing techniques adequate to indoor environments with Deep Learning techniques for Speech Processing will enable us to integrate other projects in order to improve their efficiency and will also open paths for exploring new applications for this type of technology.

1.2 Objectives

The main goal of this thesis is to discover the possibility of employing Micro Aerial Vehicles (MAV) to acquire speech audio signals and to propose methods for processing and recognition of the noisy speech signals.
To achieve this goal we set a series of objectives to our research:

- Discover and study what are the main noise reduction algorithms that have already been applied to noise reduction in the cases when a microphone is implemented in an Unmanned Aerial Vehicle (UAV).

- Explore other noise removal algorithms that are usually applied in signals with residual noise and see if they can be applied to this specific situation where, in most cases, the inputs will have negative or close to zero Signal-to-Noise Ratio (SNR).

- Find a correspondence between the motors' speeds of a drone and the filter's coefficients to be applied to the noisy inputs.

- Explore the possibility of training a Neural Network with inputs that are already corrupted by noise and see if this type of training produces good results. If so, study the competitiveness of this approach with the approach of denoising first the inputs.

1.3 Thesis Outline

This thesis is divided in 5 chapters. The present chapter is an introduction to the research that we will be developing. We start by presenting our motivation and importance of working in the topic of speech processing applied to UAV. Our motivation is followed by setting a series of objectives that we want to achieve in order to present a reliable ASR system to be implemented in UAVs.

The second chapter is divided in two main parts: Theoretical Overview and State-of-the-Art Methods. On the first part we provide a brief explanation on basic concepts of the three main topics that will be explored in this research: Automatic Speech Recognition, Deep Learning and Unmanned Aerial Vehicles. On the second part of this chapter we mention our research on what are the most relevant methods and technologies being used nowadays for these three topics.

Chapter 3 is focused on the theoretical side of the implementations developed. We start by presenting and explaining the 5 filtering algorithms that we chose to use in our implementations, followed by the description of a Feedforward Neural Network (FNN) created to predict filters' coefficients. The next topic developed is the usage of a Recurrent Neural Network that is able to receive noisy speech as input and accurately give as output the transcription of the speech. The 4^{th} section of this chapter is an introduction of the hardware and software used on the development of the ASR system and an explanation on the datasets chosen and created to test our implementations. Lastly we explain what is our criterion to classify and compare the performance of the ASR systems.

Chapter 4 presents the results of the ASR systems explained in chapter 3 and is divided in 4 sections: performance of the ASR system when the input isn't filtered, performance of the ASR system when the input is filtered, performance of the ASR system when the input is filtered and the filter's coefficients are computed through a FNN and performance of the ASR system when a Recurrent Neural Network (RNN) is trained with noisy inputs. At the end of each section an analysis of the obtained results is provided.

On the 5^{th} chapter we state our conclusions based on the work and results obtained through the research and we provide some suggestions to be explored in the future in order to improve the performance of these type of systems.

The last content of this thesis are Appendix A where we present a list of commands used to test the ASR systems and Appendix B that provides additional information on the original files of our testing commands such as SNR and Word Error Rate (WER) for each commands individually when the noise files are mixed with speech files without SNR adaptation.

2

Background

Contents

This chapter serves to present basic concepts on Speech Recognition (SR) and Deep Neural Networks (DNN). A brief background information on UAV will also be provided. Furthermore, we will make a literature review on state-of-the-art methods for noise suppression of external sources such as wind or reverberation and for ego-noise. We will also present recently explored DNN architectures for Speech Processing.

2.1 Theoretical Overview

2.1.1 Automatic Speech Recognition

Automatic Speech Recognition is the conversion of a speech signal into a sequence of words through the analysis of the signal's waveform. The complexity of an ASR system can vary according to its robustness against speech uncertainty. Type of Speech, Speaker Dependency, Dimension and Confusability of Vocabulary and Surrounding Adverse Conditions are some examples of what can be different in each sample of speech to be analyzed. The high variability of these characteristics in speech is one of the biggest barriers for the development of robust ASR systems [1].

The traditional ASR pipeline is presented in scheme 2.1:

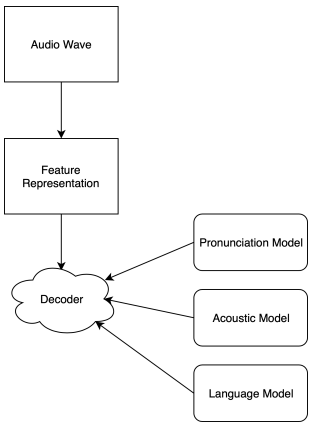


Figure 2.1: Traditional ASR system

2.1.1.A Feature Extraction

According to the scheme in Figure 2.1, the first step of the model is to extract the features of the waveform. These features can be extracted through Spectral Signal Processing (Mel-Scale Frequency Cepstral Coefficients (MFCC) [2], Perceptual Linear Prediction (PLP) [3]), Temporal Signal Processing (Delta-cepstrum) [4] or Temporal Filtering (Cepstral Mean Subtraction (CMS) [5]). The most used feature

extraction for Speech Recognition is MFCC. MFCC features are a spectral representation of phonemes with adaptation to the human auditory system by using the Mel-scale instead of linearly-spaced frequency bands. The conversion between MFCC, m, and frequency in Hertz,f, is computed through equation 2.1.

$$m = 2595 \times log_{10}\Big(1 + \frac{f}{700}\Big) \tag{2.1}$$

Before moving to Decoding, it is necessary to build the Acoustic, Lexical and Language models.

2.1.1.B Acoustic Model

Speech is the propagation of a sound wave through the vocal tract and the air. The main goal of building an Acoustic Model is to determine the Probability Density Function (PDF) of a sequence of acoustic feature vectors. In order to achieve the goal, it is necessary to consider the changes in the vocal tract, nose and lips to produce each phoneme. As it is easy to understand, trying to create a model that covers all the possibilities is a task that has not yet been fully accomplished. Until some years ago, Hidden Markov Models (HMM) [6] were the most used to estimate the acoustic model. Through time, we have seen that using Hybrid Hidden Markov Models/ Artificial Neural Networks (HMM/ANN) [7] has improved the results of Speech Recognition. Recent literature shows that using Neural Networks (NN) for the acoustic model is the path to improve future results to the point when HMM will not be used anymore [8].

2.1.1.C Pronunciation Model

In lots of different languages there are letters which can be pronounced in different ways according to its position on the word. To be able to write down what we hear it is necessary to have a dictionary which makes the conversion having in mind the issue stated. The Pronunciation Model enables the representation of each pronunciation using the basic acoustic units of the recognizer. The model has to consider the variations of pronunciation due to variances in dialects, word boundary effects and the acceptance of pronouncing some words in more than one way.

2.1.1.D Language Model

A spoken language understanding system is improved when we create a model which deals with the probability associated to the appearance of certain words after and/or before others. The Language Model is created having in consideration grammar rules and likelihood of using certain words according

to a specific context. With Stochastic Language Models (SLM) the goal is not only to improve the accuracy of the ASR system but also adding some constraints to the decoder, reducing its search space.

2.1.1.E Decoder

Once feature extraction is done and the three previous models are defined, it is time to decode all the gathered information.

The first approach used for ASR was Pattern Recognition [9] which is based in statistics. It creates probabilistic models for the patterns and, through comparison, fits the data in the model. Due to the extend of existent vocabulary and all of its possible variations it is clear that this method could only work for a small set of vocabulary and was not very robust against the uncertainties of speech.

With the introduction of HMM in Acoustic Models the goal of the decoding process is to find the most likely word $w*$ given the sequence of feature vectors O [10]. Now this is made through dynamic programming algorithms such as the Viterbi algorithm. With the Viterbi algorithm we find the best path that leads to the generation of the observation sequence. In practice, the complete search can only be done with small vocabularies. Because of this problem, there can be made some different approaches to the algorithm in order to make it more efficient. One alternative is to constrain the search space to multiple parallel hypotheses or expand it as the search progresses. The other alternative is to make a depth-first search instead of a breadth-first search as it is usually done with this algorithm.

The next big improvement of results happened when Gaussian Mixture Models (GMM) started to be used in association with HMM. GMM-HMM models were used for a long time and are still widely used methods for speech recognition. The main idea of the combination of these two models is that HMM take into account the variability of speech in time and GMM determines how good each state of the HMM represents the coefficients of the input speech signal.

In more recent years, Deep Learning (DL) has been having a very important role in exploiting new technologies and improve already existing ones. The first approach with this technology for ASR, as it has already been mentioned, was for the Acoustic Model. Although this approach showed large improvements in the outcomes of ASR, the ideal technique is to apply DL in the whole process. In the next section a resume on Deep Learning will be presented and a small explanation on Neural Networks used for Speech Recognition will also be given.

2.1.2 Deep Learning

Deep Learning [11] is a smaller set of Machine Learning methods which learn underlying features in data through Neural Networks. The main aspect of Deep Learning is that although the layers that compose

the NN are designed by humans, its parameters and features are set and adjusted by itself without human interference. A Neural Network is the collection of many simple processors called neurons [12].

2.1.2.A The Perceptron

We will start by the fundamental building block of Deep Learning: The Perceptron. This block represents a single neuron with forward propagation. Scheme 2.2 shows how this block works and it can be translated by expression 2.2.

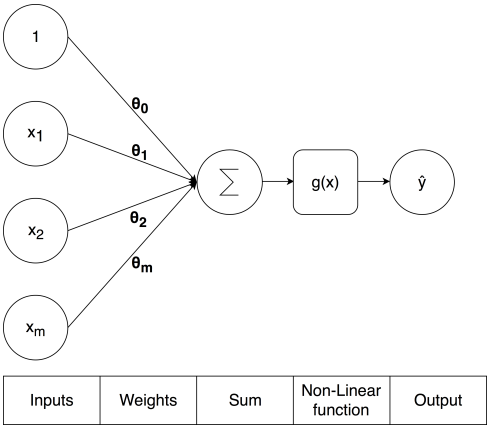


Figure 2.2: Perceptron scheme.

$$\hat{y} = g\Big(\theta_0 + \sum_{i=1}^{m} x_i \theta_i\Big) \tag{2.2}$$

The expression is composed by inputs, x_i, and weights, θ_i. $g(x)$ is a non-linear activation function. Some examples of non-linear functions that can be used are Sigmoid function, Hyperbolic Tangent and Rectified Linear Unit. The reason for the use of non-linear activation functions is to translate the non-linearities of reality into the network.

2.1.2.B Neural Networks

The next step to build a Neural Network is to add various perceptrons. We will now move one step further and explain a NN with only one hidden layer. The new scheme with the simplification can be seen in figure 2.3.

The new equations for the NN are in 2.3 and 2.4.

$$z_i = g\Big(\theta_{0,i}^{(1)} + \sum_{j=1}^{m} x_j \theta_{j,i}^{(1)}\Big) \tag{2.3}$$

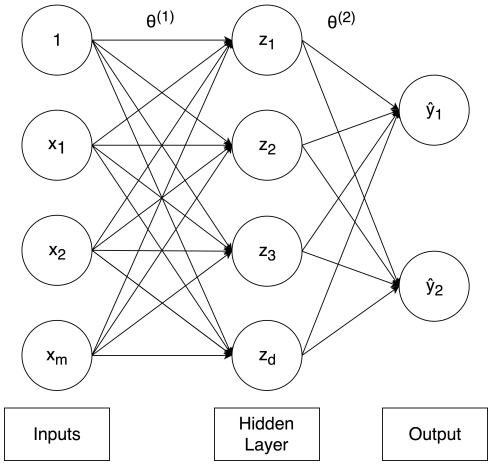


Figure 2.3: Single Layer Neural Network scheme.

$$\hat{y}_i = \left(\theta_{0,i}^{(2)} + \sum_{j=1}^{d} z_j \theta_{j,i}^{(2)}\right) \tag{2.4}$$

So, as it can be seen in the scheme, we have now one hidden layer. This layer provides results that are not necessarily relevant for the output. Each layer has different weight matrices, θ_1 and θ_2, and these weights are the ones responsible for the different outcomes. In order to have a fully connected neural network we just need to add hidden layers and connect all the nodes between each layer.

2.1.2.C The Concept of Loss

After having the NN we cannot start using it right away and expect good results. The first part of using a NN is the training. Training is done through the computation of the difference between the result we predict, $f(x^{(i)};\theta)$,and the actual result, $y^{(i)}$. The loss of a network is the cost associated to an incorrect prediction, $\mathcal{L}(f(x^{(i)};\theta), y^{(i)})$. If we want to compute the loss associated to an entire dataset we have to compute the empirical loss through the cost function in 2.5.

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(f(x^{(i)};\theta), y^{(i)}) \tag{2.5}$$

Where n is the number of hidden layers.

The Mean Squared Error (MSE) cost function described in equation 2.6 is an example of a cost function that can be used when we have more than one output.

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} (y^{(i)} - f(x^{(i)};\theta))^2 \tag{2.6}$$

2.1.2.D Training of a Neural Network

After knowing the concept of loss, it is easy to understand how it can be integrated in the training of a NN. The weights, θ, will be computed through the minimization of the cost function (equation 2.7). The optimal cost can be found by computing the gradient descent.

$$\theta^* = \underset{\theta}{\operatorname{argmin}} J(\theta) \tag{2.7}$$

In order to find the weights of each layer we have to make Backpropagation. Figure 2.4 and equations 2.8 and 2.9 translate the steps necessary to compute the optimal weights in the simplest case of a NN with one hidden layer.

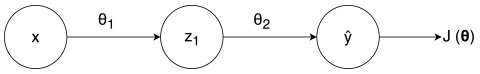


Figure 2.4: Simplified Single Layer Neural Network scheme.

$$\frac{\partial J(\theta)}{\partial \theta_2} = \frac{\partial J(\theta)}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial \theta_2} \tag{2.8}$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = \frac{\partial J(\theta)}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_1} \times \frac{\partial z_1}{\partial \theta_1} \tag{2.9}$$

Actually the explained method has high computational costs. In order to reduce these costs what is done is choosing a smaller set of points and use those data points to compute an average gradient. This is the most simple type of Neural Network. We are now going to provide a small explanation on other two types of NN.

2.1.2.E Convolutional Neural Networks

When we have data that comes in the form of arrays, like what happens with images, usually the type of NN used is a Convolutional Neural Network (CNN) [11] represented in Figure 2.5.

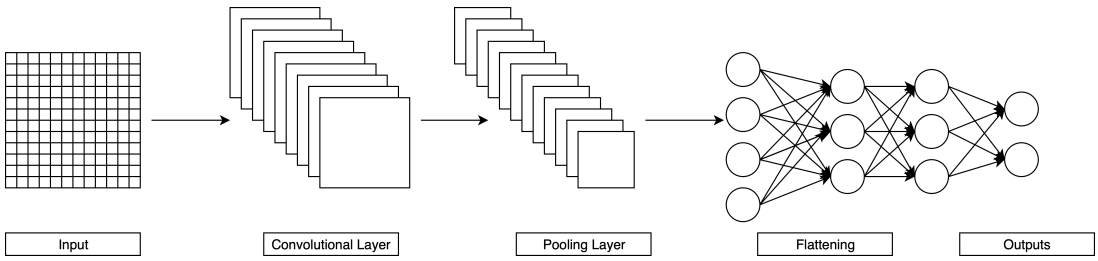


Figure 2.5: Convolutional Neural Network scheme.

A CNN has multiple stages being the first ones composed by convolutional and pooling layers. The

best example to explain how a CNN works is using image processing. An image can be translated to a matrix of pixel values. The first step of the CNN is to apply a filter of the desired size. The weights in the filter will allow the extraction of local features. Let's assume we choose a 3x3 filter. What will happen is that the layer will make the convolution of the pixels with the filter and will store the result in another matrix as it can be seen in Figure 2.6.

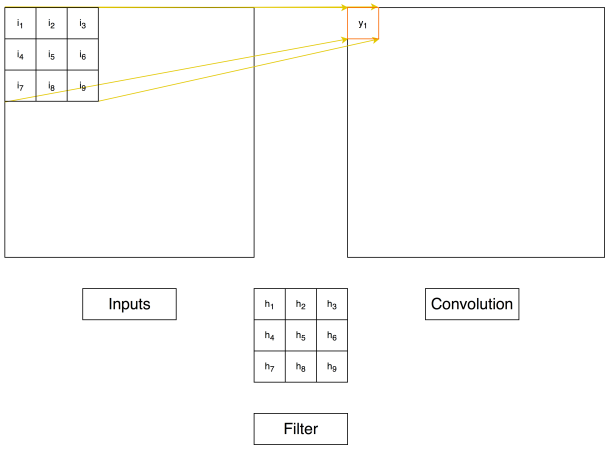


Figure 2.6: Convolution scheme.

Changing the weights of the filter allows the detection of different patterns. As we progress on the layers, the filters tend to detect more complex patterns. The next step is to apply the non-linearity mentioned in the explanation of NN. One of the functions that can be used after each convolution is the Rectified Linear Unit which replaces the negative values by zero. The last step on a CNN is the Pooling operation. It is used to reduce the dimension of the output. A patch is chosen and, the most common technique used to choose a value is max pooling, the highest value of the patch is chosen for the output.

2.1.2.F Recurrent Neural Networks

The other type of NN that can be used are RNN [13] presented in Figure 2.8. This scheme is based on Figure 2.7 which is a simplification of Figure 2.3 with the addition of one hidden layer.

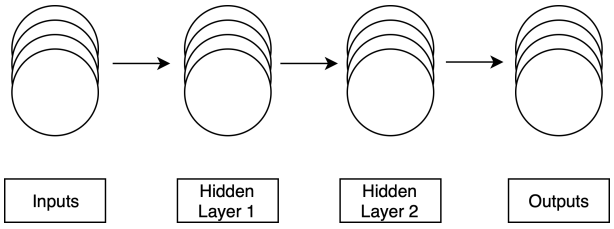


Figure 2.7: Simplification of FNN scheme.

These types of NN have shown to work better in Speech Processing tasks. One of the differences of these networks is that instead of having a single input single output data, we can have flexibility on the

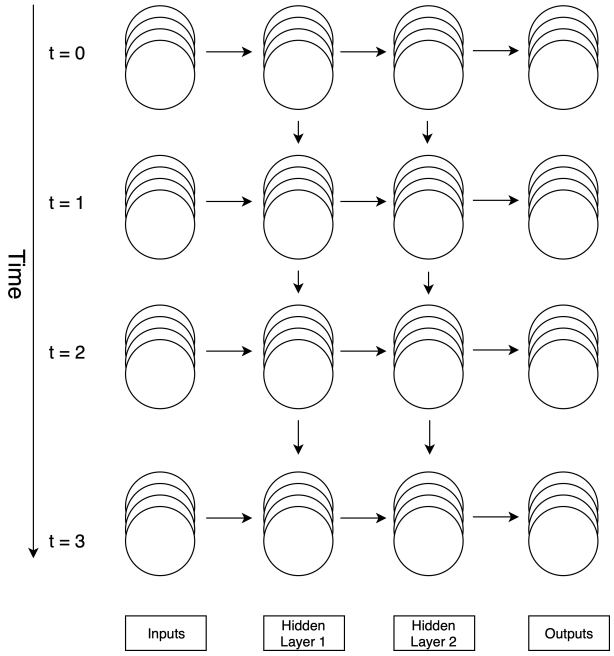


Figure 2.8: Recurrent Neural Network scheme.

quantity of inputs and outputs of the system. In RNN each element of an input is processed at a time and a vector containing information about the history of past elements of the sequence is maintained. The goal is to keep information for long time so we can have more data to predict the output word but storing so much information is very difficult. Long-Short Term Memory (LSTM) networks [14] deal with the problem mentioned and have shown better results than conventional RNN. Other proposed solutions are the Neural Turing Machine [15] and Memory Networks [16]. All of these methods find a way of increasing the memory of the network.

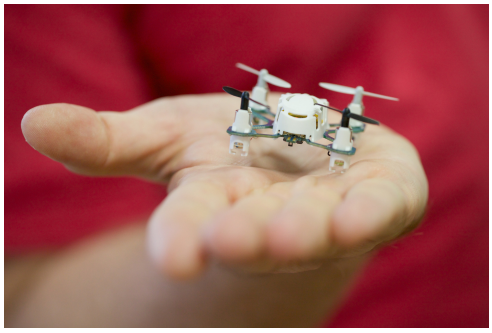
2.1.3 Unmanned Aerial Vehicles

UAVs are aircrafts which have no human pilots aboard. These vehicles are most commonly known as drones. Their first appearance dates back to 1894 when Austria attacked the Italian city of Venice with unmanned balloons full of explosives. On the next century, UAVs became more sophisticated mainly due to military purposes. The First and Second World Wars enabled a big development of this technology. With remotely controlled aircrafts they could attack other cities without human losses on their side. These UAVs had considerable dimensions to transport the wanted load and have the needed fuel to travel the necessary distances. In figure 2.9 we can see a United States Navy radio-controlled aircraft of the World War I.

In the 1990's miniature and micro UAVs started being produced. In this updated version of drones

Figure 2.9: Curtiss N2C-2 drone.

the size was extremely reduced. It is important to keep in mind that the autonomy is also small. These first small drones still had a high cost so they would be acquired by companies specially to capture images. Only in recent years drones started to be commercialized and accessible to consumers. Future perspectives show that drones will not only be used to capture images but also sound and, besides capturing data, they will be able to carry heavier weights and make delivers. Images 2.10 show some examples of UAVs nowadays.



(a) Micro UAV

(b) Amazon Prime Delivery Drone

Figure 2.10: Drones nowadays.

2.2 State-of-the-art Methods

In this section we will present what are the state-of-the-art methods of each section explored in the Theoretical Overview.

2.2.1 Signal Enhancement and Filtering for Automatic Speech Recognition

As it was mentioned in subsection 2.1.1, recent literature is focusing on the advantages of using uniquely Deep Learning for ASR. In order to improve the performance of an ASR system and reduce its computational cost the inputs should be as clear as possible. What this means is that a noisy speech signal should be enhanced or filtered so we can feed a Neural Network an input containing speech-only. Some filtering algorithms use the Fourier Transform (FT) to make a frequency analysis. The main problem is that we cannot make an analysis of the temporal variations in the frequency spectrum. Short-Time Fourier Transform (STFT) appeared to overcome this problem. To also be able to perform a time-analysis of the signal, a fixed-length window is applied through consecutive portions of the signals and the FT is continuously perform. Most of the filtering algorithms used for signal denoising work with the STFT.
In the next subsections we will focus on a series of different approaches to reduce all kinds of noise that can corrupt a speech signal captured by a drone.

2.2.1.A Beamforming

If we think about a drone flying in an open space area, localizing the speech source and focusing the capture of speech in that direction can be considered as one method of speech enhancement.
This is exactly what is achieved through Beamforming. Beamforming is a signal processing method where a multi-channel microphone system is used to form a spatial filter that can extract signals from a specific direction while reducing the effect of signals coming from other directions. As an example of Beamforming we find the General Sidelobe Canceller (GSC) algorithm [17].

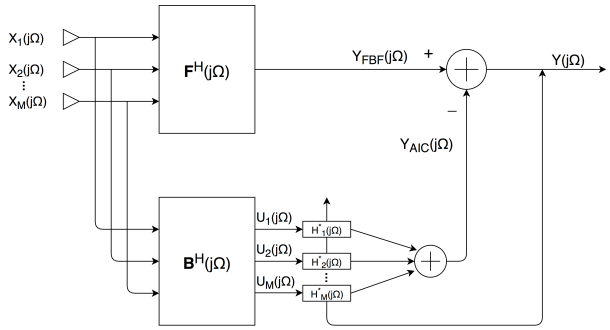


Figure 2.11: Block Diagram of the GSC.

The block diagram in Figure 2.11 can be translated to the equation 2.10 where $\mathbf{F}$ is the fixed beamformer, $\mathbf{H}$ is the adaptive interference canceller and $\mathbf{B}$ is the blocking matrix. k and p are the frequency bin and time-frame, respectively.

$$\mathbf{W}(k,p) = \mathbf{F}(k,p) - \mathbf{H}(k,p)\mathbf{B}(k,p) \tag{2.10}$$

This algorithm is a multi-channel signal enhancement approach that allows the elimination of sound signals coming from unwanted sources. In [17] there is also explored some small alterations to the GSC which will improve the ASR results.
Signal processing through beamforming allows us to have a signal with less interferences from background noise for further filtering.

2.2.1.B Reverberation

One of the main challenges for ASR in an indoor environment is Reverberation. Since we are in a confined area the sound waves produced by the source will travel through the room and reflect on the walls. The microphones in the drones will capture not only the signal but also its reflections.
Based in [18] we were able to find what is thought to be the best and most recent methods to solve the problem stated before. The first methods referred are STFT - domain inverse filtering. The idea of these methods is based on the fact that the wanted clean speech is not correlated with the sound produced by the reverberation due to the time-varying nature of speech. The first step of the algorithm is to apply a Weighted Linear Prediction Error method. Through a prediction error vector and a prediction matrix multiplied by past STFT coefficients of microphones' signals, we can obtain the coefficients' vector of the present time frame, $y_n[f]$. To obtain the dereverberated signal it is just necessary to subtract the late reverberant component (part of the signal with zero correlation) to $y_n[f]$.
The other two methods presented to remove reverberation are based on Room Impulse Response (RIR) and both work with mono-channel signals. The first one is based on the idea that the observed signal, $Y_n[f]$, is the convolution of the clean speech signal, $S_n[f]$, with the RIR, $H_m[f]$, as it can be seen in equation 2.11

$$Y_n[f] = \sum_{m=1}^{M-1} S_{n-m}[f] H_m[f] \tag{2.11}$$

In the equation, f represents the frequency bin and n the frame. The common approach to find matrices $S_{n-m}[f]$ and $H_m[f]$ is to use the Non-negative Matrix Factor Deconvolution (NMFD). $Y_n[f]$ is decomposed into the convolution of the RIR, $H_m[f]$ and the clean speech, $S_{n-m}[f]$. The particularity of this algorithm is that all the entries of the matrices are non-negative. The fact that this algorithm only corrects amplitude makes it more robust than the previous one which also attempts to correct the phase information of the signal.
The other model based on RIR modeling states that the RIR, $h(t)$, is a white noise modeled by a decaying exponential whose decaying rate is defined by the reverberation time. A prediction of the reverberation's power spectrum can be done and then this prediction is subtracted to the observed power spectrum. We are then able to obtain a prediction of the dereverberated speech signal.

Other methods to remove reverberation can be found in the same article [18] and in [19].

2.2.1.C Noise Removal

Filtering noise is a complex task and the one which gives us the possibility to explore more paths and ideas to improve the quality of speech signals. There are several techniques to remove unwanted sounds from the signal. In [20] the method proposed is Asymmetric Noise Suppression. This method bases on the idea that speech power has higher and quicker changes than background noise power. An envelope of the lower power level is computed and is subtracted to the signal. This way, we can obtain an approximation of the speech-only signal.

The other way of reducing the effect of noise in ASR systems is to perform speech enhancement instead of completely removing the noise of the signals (the previously explained method of Beamforming is a good example of speech enhancement). In [21] the signal, $y[k]$, is described as $y[k] = x[k] + n[k]$, the sum of the clean speech and corrupting noise, respectively. Doing the computations in the frequency domain and using non-linear spectral subtraction, the algorithm computes the SNR and, for a high value, subtractions for spectral components are reduced and, for a low value of SNR, subtractions for spectral components are increased. The background noise is never completely removed but the SNR is improved.

All of these methods are important for background noise removal but our greatest concern is to decrease the impact of the main source of noise in our project, the noise of the drones' motors. We will explore more on this topic in subsection 2.2.3.

2.2.2 Deep Learning

The implementation of Neural Networks is already a well established approach for Speech Recognition. Nowadays new NN architectures are being explored with the aim of improving the predictions made by the NN and reducing the complexity of the systems. One of the approaches that has been studied is the usage of CNN. CNN are usually implemented for image processing and its performance has proven to be very satisfactory. In [22] we find an approach that treats the speech signal's spectrogram as an image and through that assumption is able to perform speech recognition.

In [23] the goal of their approach is to significantly reduce the computational costs associated to the implementation of a NN. For speech recognition if the vocabulary is reduced, the training and usage of DL can be done in a personal computer in a small set of hours but if the training data is very large, all the processing can take up to days. In [23] the idea of a Partially-Shared Deep Neural Network (PS-DNN) is explored. Using PS-DNN, it is possible to learn two different tasks starting from the same input. Figure

2.12 shows the structure of the network.

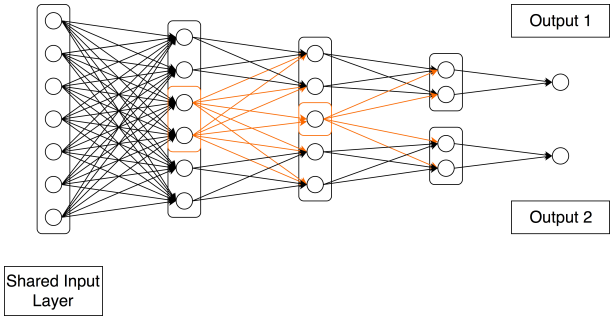


Figure 2.12: Structure of the PS-DNN.

In the figure we can see that the input layer is fully connected to the first hidden layer. What happens in the next layer is that we will have a part associated to one task, a part common to both tasks (in orange) and a part associated to the other task. It is possible to do this because the independent part that belongs to one sub-network will not influence the upper layers of the other sub-network. It is important to note that the ratio of the shared and independent parts vary according to the tasks. On the last hidden layer we can see that is even possible the disappearance of the common part. At the end we have the two different outputs of the two different tasks.

There are toolboxes for several programs which allow us to implement a Neural Network. For example to program with Matlab there is the Deep Learning toolbox. Microsoft also provides the Microsoft Cognitive Toolkit to train DL algorithms which can be used in Python, C and C++ languages. The most used open source libraries of machine learning are TensorFlow (Google) and PyTorch (Facebook). With these libraries we are able to implement and use our NN.

2.2.3 Unmanned Aerial Vehicles

Nowadays UAVs can have such small measures that they can even fit in a person's hand. The fact of having drones with such small size leads to small engines and rotors. As it can be easily understood, these smaller components don't produce as much noise as the ones we can find in big UAVs. Nevertheless, it is necessary to think that by using such small drones, the microphone or the array of microphones is going to be implemented very close to the source of noise. The noise power coming from the drone can easily overcome the power of the speech signal leading to the failure of the whole system.

In recent years the ego-noise cancelling problem in drones has been studied with the aim of contributing for better performances of sound source localization systems. One of the proposed algorithms for sound source localization that also deals with noise is the MUltiple SIgnal Classification based on incremental Generalized Singular Value Decomposition (iGSVD-MUSIC) [24]. This algorithm allows the localization

of sound sources by whitening high power noise using a noise correlation matrix which is incrementally and adaptively estimated.

The article [25] specifically addresses the problem of ego-noise reduction for small UAV and presents a study on three different types of filtering: the already mentioned Beamforming, Blind Source Separation and Time-Frequency Processing. This last one has shown to have a better performance than the other two. The main idea of this algorithm is the estimation of Direction of Arrival of the speech at each time-frequency bin (obtained through the STFT) and then combine the localization results from individual time-frequency bins to perform noise reduction.

The ego-noise reduction or cancelling is still an area that offers lots of opportunities for research.

3

Implementation

Contents

In this chapter we will start by explaining the algorithms tested for noise filtering followed by the small explanation of a Neural Network trained with noisy speech for ASR that was also tested. The implementations are followed by the presentation of the devices used in this project. We will also explain how the dataset for testing was created and how the tests' classification is done.

3.1 Filtering Algorithms

The main obstacle to be overcome in this project is the speech enhancement. The problem of noise coming from external sources has already been addressed in the two previous chapters and some solutions can be found in the methods mentioned in the State-of-the-art section. In our research we want to focus on the topic that we believe that has not been widely explored yet, ego-noise reduction in UAVs. The topic of ego-noise reduction has had some contributions such as [26] and [27] but for both these works the study was developed in ground robots. We want to understand if the fundaments of the solutions of the problem are valid for drones despite their very loud self engine noise and if classical methods for noise removal/reduction can be a good contribution to solve this specific problem.

In this work we will implement and compare five different filtering methods to reduce or remove the motors' noise. The first two methods implemented are the most popular ones, Wiener Filter and Least-Mean-Square Adaptive Filter. The third method chosen is the one applied in the program Audacity [28] for noise removal, Noise Gate. As for the forth developed filter we implemented a method proposed in [27] called Motion-Dependent Spectral Subtraction. Finally, we also explored the Discrete Wavelet Transform for noise removal.

3.1.1 Wiener Filter

Applying a Wiener Filter to a noisy signal is a very popular technique for speech enhancement in signal processing. This approach is based on Minimum Mean Squared Error (MMSE) between the desired signal and an estimation of that desired signal. To use the Wiener Filter we have to consider the discrete noisy signal, $x[n]$ as in 3.1 where the noise, $n[n]$, is independent and additive regarding to the speech signal, $s[n]$.

$$x[n] = s[n] + n[n] \tag{3.1}$$

In order to obtain an estimation of the clean speech we will follow the methods used in [29]. We will

start by applying the Two-Step Noise Reduction Technique (TSRT) followed by the speech Harmonic Regeneration Noise Reduction (HRNR). The first step of the algorithm is to apply the STFT to the noisy speech. Let $X(k,p)$, $S(k,p)$ and $N(k,p)$ be the k-th spectral component of the short-time frame p of the noisy signal, speech and noise, respectively. The evaluation of $a\ priori$ SNR and $a\ posteriori$ SNR is typically used for the evaluation of the efficiency of the speech enhancement. The equations to compute these two values of SNR are given by 3.2 and 3.3.

$$S\hat{N}R_{prio}(k,p) = \frac{E\left[|S(k,p)|^2\right]}{E\left[|N(k,p)|^2\right]} \tag{3.2}$$

$$S\hat{N}R_{post}(k,p) = \frac{|X(k,p)|^2}{E\left[|N(k,p)|^2\right]} \tag{3.3}$$

where $E[.]$ represents the expectation operator. The spectral gain can be then obtained through equation 3.4 in which function $g(.)$ is a gain function proposed in methods like power spectral subtraction or wiener filter.

$$G(k,p) = g(S\hat{N}R_{prio}(k,p), S\hat{N}R_{post}(k,p)) \tag{3.4}$$

In the frequency domain we can now obtain the predicted speech signal $\hat{S}(k,p)$ by applying the gain function $G(k,p)$ to the original signal $X(k,p)$ as in equation 3.5

$$\hat{S}(k,p) = G(k,p)X(k,p) \tag{3.5}$$

Through the reading of [29] we conclude that the computation of the minimum of $E\{(\hat{S}(k,p) - S(k,p))^2\}$ leads to equation 3.6

$$G(k,p) = \frac{E\left[|S(k,p)|^2\right]}{E\left[|S(k,p)|^2\right] + E\left[|N(k,p)|^2\right]} = \frac{S\hat{N}R_{prio}(k,p)}{1 + S\hat{N}R_{prio}(k,p)} \tag{3.6}$$

Looking at equation 3.2 we realize that from the original signal we can only obtain $X(k,p)$. $S(k,p)$ and $N(k,p)$ are unknown matrices. To compute the value of $S\hat{N}R_{prio}(k,p)$ we apply now the Decision Direct Approach (DD). With this approach, the $S\hat{N}R^{DD}_{post}(k,p)$ and the $S\hat{N}R^{DD}_{prio}(k,p)$ are as in the following equations:

$$S\hat{N}R_{post}(k,p) = \frac{|X(k,p)|^2}{\hat{\gamma}_n(k,p)} \tag{3.7}$$

$$S\hat{N}R_{prio}^{DD}(k,p) = \beta\frac{|\hat{S}(k,p-1)|^2}{\hat{\gamma}_n(k,p)} + (1-\beta)P\left[S\hat{N}R_{post}(k,p) - 1\right] \tag{3.8}$$

where β is a value between 0 and 1 which controls the behaviour of the decision made by the algorithm (usually $\beta = 0.98$), $\hat{S}(k,p-1)$ is the estimation of the speech spectrum at the previous frame, $\hat{\gamma}_n(k,p)$ is the estimation of the noise Power Spectral Density (PSD) through minima controlled recursive averaging approach [30] and $P[.]$ represents the half-wave rectification.

Once we have obtained G_{DD} through equation 3.6 we can now advance to the TSRT method to make the prediction of speech. The new $a\ priori$ SNR is given by

$$S\hat{N}R_{prio}^{TSRT}(k,p) = \frac{|G_{DD}(k,p)X(k,p)|^2}{\hat{\gamma}_n(k,p)} \tag{3.9}$$

Applying the result of equation 3.9 to equation 3.6, the estimation of the clean speech can finally be computed through equation 3.10.

$$\hat{S}(k,p) = G_{TSRT}(k,p)X(k,p) \tag{3.10}$$

Listening to the predicted speech obtained with the TSRT method it is clear that a distortion is present in the signal. This distortion is the result of estimation errors along the previous method. What happens is that some harmonics are considered as components of only noise and are removed in the filtering process. This harmonic distortion can be overcome when we implement the HRNR method. The principle of this implementation is that when we apply a non-linear function (that in this case is the maximum relative to 0) to the predicted speech signal in the time domain, $\hat{s}(t)$, we obtain the harmonics at the same positions as in the clean speech. Since these harmonics have biased amplitudes, we cannot directly use them to restore the speech signal but we can use them to improve the wanted signal. The final estimation of the clean speech with restore of the harmonics is obtained applying equations 3.11 to 3.14.

$$s_{harmo} = NL(\hat{s}(t)) = max(\hat{s}(t), 0) \tag{3.11}$$

$$S\hat{N}R_{prio}^{HRNR}(k,p) = \frac{G_{TSRT}(k,p)|\hat{S}(k,p)|^2 + (1 - G_{TSRT}(k,p))|S_{harmo}(k,p)|^2}{\hat{\gamma}_n(k,p)} \tag{3.12}$$

$$G_{HRNR}(k,p) = \frac{S\hat{N}R_{prio}^{HRNR}(k,p)}{1 + S\hat{N}R_{prio}^{HRNR}(k,p)} \tag{3.13}$$

$$\hat{S}(k,p) = G_{HRNR}(k,p)X(k,p) \tag{3.14}$$

3.1.2 Least-Mean-Square Adaptive Filter

Another type of filter that can be used for noise suppression is the Least Mean Square (LMS) Adaptive Filter [31]. This kind of filter is based in a widely used algorithm, the LMS. Unlike the Wiener Filter, there is no need to measure correlation functions which makes the LMS a simple although robust algorithm.

The LMS adaptive filtering algorithm consists in two steps:

- Filtering Process: Passing an input signal by a linear filter and compare the result with a desired response. This comparison originates an estimation of error.
- Adaptive Process: Adjusting the parameters of the linear filter in order to reduce the estimated error.

In figure 3.1 we have the block diagram of an adaptive filter that translates the steps mentioned before.

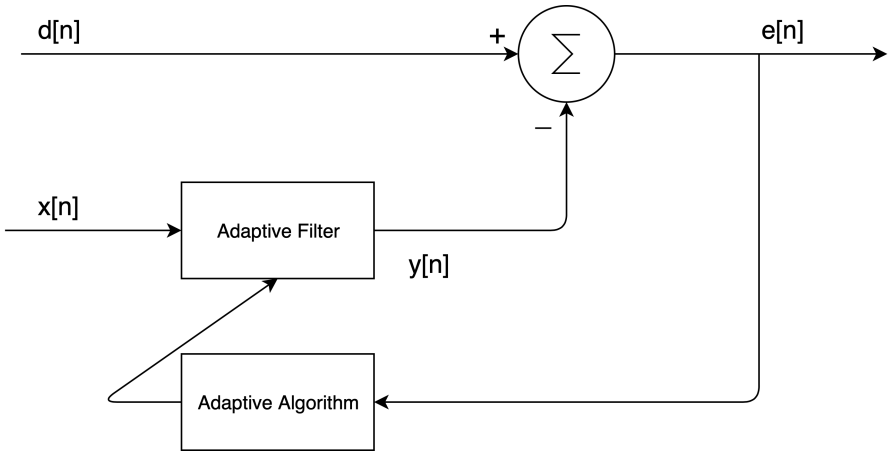


Figure 3.1: Block diagram of the adaptive filter.

In the figure $x[n]$ is the noisy input signal, $y[n]$ is the filtered signal, $d[n]$ is the desired signal and $e[n]$ is the estimated error which is the result of the subtraction between $d[n]$ and $y[n]$. The goal of the algorithm is to repeat the process until the error is 0 which means a match between the output signal and the desired signal.

Specifically for this project what we want to implement is Adaptive Noise Cancelling. Figure 3.2 shows the block diagram of the algorithm that will be implemented.

In this case, the inputs to the algorithm are the noisy signal ($x[n] = s[n] + n[n]$) and noise-only signal ($n[n]$). What happens is that the noise-only input is only correlated to the component of noise

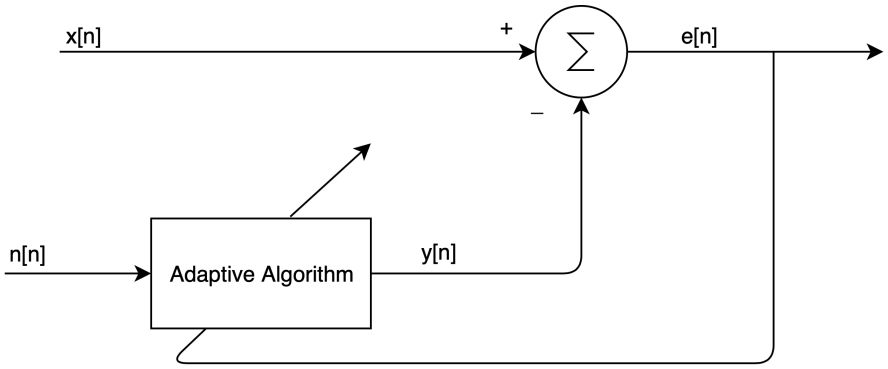


Figure 3.2: Block diagram of Adaptive Noise Cancelling.

of the noisy signal so the filtering step is only able to approximate the output of the filter, $y[n]$, to the noise component of $x[n]$. Having this in mind, in order to obtain an estimation of the clean speech, $\hat{s}[n]$, we have to consider that the desired signal is $x[n]$ and by looking at the diagram in figure 3.2, we can write that $e[n] = x[n] - y[n]$ and conclude that $e[n] = \hat{s}[n]$. The algorithm used was Matlab's dsp.LMSFilter [32]. Equations 3.15 to 3.20 show the computations required to obtain the prediction of the speech-only signal.

$$y[n] = \boldsymbol{w}^T[n-1]\boldsymbol{u}[n] \tag{3.15}$$

$$\boldsymbol{u}[n] = [n[n] \quad n[n-1] \quad ... \quad n[n-order]] \tag{3.16}$$

$$\boldsymbol{w}[n] = \alpha\boldsymbol{w}[n-1] + f(\boldsymbol{u}[n], e[n], \mu) \tag{3.17}$$

$$f(\boldsymbol{u}[n], e[n], \mu) = \mu e[n] sign(\boldsymbol{u}[n]) \tag{3.18}$$

$$sign(\boldsymbol{u}[n]) = \begin{cases} -1 & \text{if } \boldsymbol{u}[n] < 0 \\ 0 & \text{if } \boldsymbol{u}[n] = 0 \\ 1 & \text{if } \boldsymbol{u}[n] > 0 \end{cases} \tag{3.19}$$

$$e[n] = x[n] - y[n] \tag{3.20}$$

where $\boldsymbol{w}[n]$ are the coefficients estimated of the filter at time n, α is the leakage factor ($0 < \alpha \leq 1$) and μ is the adaptation step size. There are some parameters that have to be defined before running

the algorithm. The first choice is already present in the previous equations. In Matlab there are five different options to compute $f(\boldsymbol{u}[n], e[n], \mu)$. We chose Sign-Data LMS that is translated in equation 3.17. The choice was obvious when listening to the results of all the different algorithms, Sign-Data LMS presented better noise removal with less speech distortion. The other parameters to be defined are the filter's order and the value of μ. We will evaluate the performance of the ASR system for the combinations of these two parameters in which $\mu \in \{0,01; 0,05; 0,075; 0,1; 0,5\}$ and the filter order $\in \{2, 5, 7, 12, 20\}$ for a low-pass filter.

3.1.3 Noise Gate

Noise Gate [33] is an algorithm that uses Fourier analysis. The first step is to apply the STFT with a Hanning window to a segment of sound that only contains noise, $N(k,p)$. After obtaining the power spectrum of the noise, we convert the power to Decibel (dB) and we can retrieve a series of parameters that describe the noise according to its frequency and time. These two steps are described in equations 3.21 and 3.22.

$$N(k,p) = STFT\{n[n]\} \tag{3.21}$$

$$N_{dB}(k,p) = 10log_{10}(N(k,p)) \tag{3.22}$$

From the previously mentioned parameters we will define a value for threshold. We start by computing the mean power along every frequency, $\bar{N}(k)$, and then we compute the standard deviation of the mean power along every frequency, σ_k. In this case, we define the threshold, λ, as in the equation 3.23.

$$\lambda(k) = \overline{N_{dB}}(k) + n \times \sigma_k \tag{3.23}$$

After having the threshold values for each frequency we can now compute the power spectrum in dB of the whole noisy signal, $x[n]$, as it was done in 3.21 and 3.22. On this step of the algorithm, a comparison between the power of the noisy signal and the threshold at each frequency and time will be done. Equation 3.24 is then applied.

$$G_{dB}(k,p) = \begin{cases} 0dB & \text{if } X_{dB}(k,p) \geq \lambda(k) \\ min(X_{dB}(k,p)) & \text{if } X_{dB}(k,p) < \lambda(k) \end{cases} \tag{3.24}$$

If the power of the signal is above the threshold, we set a gain of $0dB$ (which is translated to a non-change of the original signal). If the power is below the threshold, we set a negative gain equal to the minimum value of power that we find in the noisy signal. In order to avoid abrupt changes of power in the signal, the next step of the algorithm is to apply a smoothing filter over frequency and time, $\tilde{G}_{dB}(k,p)$.

Through 3.25 we obtain the predicted clean speech in dB.

$$\hat{s}_{dB}(k,p) = X_{dB}(k,p) + \tilde{G}_{dB}(k,p) \tag{3.25}$$

The denoised signal, $\hat{s}[n]$, is recovered by applying the inverse of equation 3.22 to go back to linear and then by computing the Inverse Short-Time Fourier Transform (ISTFT).

3.1.4 Motion-Dependent Spectral Subtraction

In [27] we find a filtering approach based in spectral subtraction like in the method explained in 3.1.3. The difference in this case is that we divide the frequencies in l sets and an estimation of noise is made for each set instead of estimating a threshold along time.

The approach followed in [27] seemed to be a good approach for noise removal but we understood it wouldn't be possible to only filter the signal in terms of its magnitude. Filtering only magnitude would never allow the recovery of the speech signal because the phase information would be lost. The following method is based on the one proposed in [27] but with slight changes in order to recover the signal. For the first part of the algorithm we have to consider a signal containing noise-only. The algorithm works in the frequency domain so first we have to compute the Discrete Fourier Transform (DFT), $N(f)$, and apply a triangular window according to equation 3.26 to divide the noise components in each set of frequencies.

$$m(l) = \sum_{f=f_{lo}(l)}^{f_{hi}(l)} W_l(f) N(f) \tag{3.26}$$

The window applied is defined in equation 3.27.

$$W_l(f) = \begin{cases} \frac{f - f_{lo}(l)}{f_c(l) - f_{lo}(l)} & \text{if } f_{lo}(l) \leq f \leq f_c(l) \\ \frac{f_{hi}(l) - f}{f_{hi}(l) - f_c(l)} & \text{if } f_c(l) \leq f \leq f_{hi}(l) \\ 0 & \text{otherwise} \end{cases} \tag{3.27}$$

In equations 3.26 and 3.27 $f_{lo}(l)$, $f_{hi}(l)$ and $f_c(l)$ represent the lowest, highest and central frequencies in the l^{th} set.

The assumption made in this algorithm is that the noise is uniform within each set of frequencies. The average noise spectrum, $\overline{N}_l$ can be then obtained by:

$$\overline{N}_l = \frac{m(l)}{\sum_{f=f_{lo}(l)}^{f_{hi}(l)} W_l(f)} \tag{3.28}$$

Once we have estimated the average noise within each set of frequencies, we can apply speech enhancement to the noisy signal $x[n]$. Once again we compute the DFT of the noisy signal, $X(f)$, and

the magnitude of the speech spectrum is computed through equation 3.29

$$\tilde{S}_l(f) = \begin{cases} X(f) - \bar{N}_l & \text{if} |X(f)|^2 - \alpha|\bar{N}_l|^2 \geq \beta \bar{N}_l^2 \\ 0 & \text{if} |X(f)|^2 - \alpha|\bar{N}_l|^2 < \beta \bar{N}_l^2 \end{cases} \tag{3.29}$$

where $\alpha \geq 1$ and $0 < \beta < 1$. To recover the speech-only spectrum we have to apply again the window computed in 3.27 to the predicted speech obtained in 3.29.

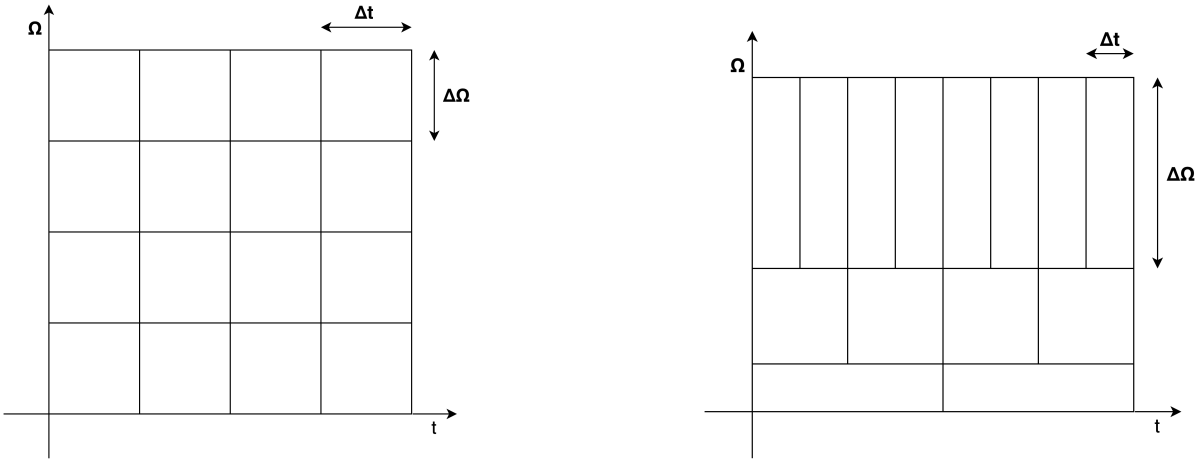
$$S(f) = \frac{\sum_{l=1}^{l\_max} W_l(f)\tilde{S}_l(f)}{\sum_{l=1}^{l\_max} W_l(f)} \tag{3.30}$$

where l_max corresponds to the number of sets that the frequencies were divided in.

The last step to recover the speech signal is to apply the Inverse Discrete Fourier Transform (IDFT).

3.1.5 Discrete Wavelet Transform

The last filter that was tested is based on the Discrete Wavelet Transform (DWT) [34]. Most of the previously addressed filters use either the FT or the Short-Time Fourier Transform. The main problem with the STFT is that we have an uniform time-frequency distribution over the time-frequency plane, as it can be seen on the left side of figure 3.3. The reality is that for low-frequency analysis we require a long time-domain and short frequency-domain window. For high-frequency analysis is the opposite, we require a short time-domain and long frequency-domain window. This is translated in the representation of the time-frequency plane on the right side of figure 3.3.



(a) Time-frequency plane for Fourier Transform

(b) Time-frequency plane for Wavelet Transform

Figure 3.3: Time-frequency planes for different types of transforms.

In order to use the DWT for audio denoising we have to follow the steps in equations 3.31 to 3.35.

$$x(t) = s(t) \otimes n(t) \tag{3.31}$$

$$y = W(x) \tag{3.32}$$

$$z = D(y, \lambda) \tag{3.33}$$

$$\hat{s} = W^{-1}(z) \tag{3.34}$$

In this set of equations $x(t)$ is the noisy signal which in the time domain is the convolution of speech, $s(t)$, with noise, $n(t)$. $W(.)$ and $W^{-1}(.)$ represent the Wavelet Transform and Inverse Wavelet Transform, $D(.)$ is a denoise operation using a threshold λ and $\hat{s}$ is the predicted speech after applying the algorithm.

When we have a discrete signal, the first step to compute the DWT is to pass the noisy signal, $x[n]$, through sets of high-pass and low-pass filters. The level of decomposition represents through how many sets of filters we pass our signal by. Figure 3.4 shows the process of a 3 level decomposition.

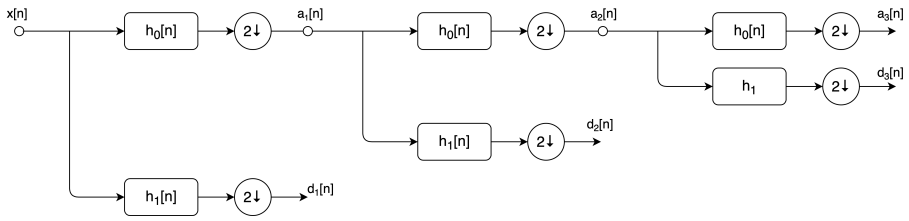


Figure 3.4: Level 3 Wavelet Transform.

In the scheme, $h_0[n]$ is the impulse response of a low-pass filter, $h_1[n]$ is the impulse response of a high-pass filter, a_n is the approximation coefficient at level n and d_n is the detail coefficient at level n. The block $2 \downarrow$ represents a downsample by 2 since after each filter the signal has less half of the frequencies. In order to define the filters h_0 and h_1 we have to go back to the definition of the DWT as it is explained in [35]. The DWT is defined in equation 3.35:

$$DWT = \int_{-\infty}^{+\infty} x(t)\psi_{j,k}(t) \tag{3.35}$$

where

$$\psi_{j,k}(t) = \frac{1}{\sqrt{s^j}}\psi\left(\frac{t - k2^j}{2^j}\right) \tag{3.36}$$

being j the scale parameter and k the shift parameter. We can conclude now that $\psi_{j,k}(t)$ is a dilated

and translated version of the mother wavelet $\psi(t)$. There are several mother wavelet functions that vary on scaling and wavelet definition (e.g. Haar, Coiflets, Symlets, Shannon and etc.). For our case we opted to choose Daubechies mother wavelet function according to results in [35]. After this process we finally obtain the values of $a_n, d_1, d_2, ..., d_n$.

The next step of the algorithm is to perform the denoising through the threshold of the detail coefficients. More detailed explanation on how to compute the threshold value will be delivered in section 4.2.5. To perform the noise removal we applied the soft threshold described in equation 3.37. We chose soft threshold instead of hard threshold because we are dealing with an audio signal.

$$z = \begin{cases} sign(x)(x - \lambda) & \text{if } x > \lambda \\ 0 & \text{if } x \leq \lambda \end{cases} \tag{3.37}$$

Finally the last step is to reconstruct the signal performing the Inverse DWT through the original approximation coefficient and the estimated detail coefficients. Figure 3.5 shows how the reconstruction is done.

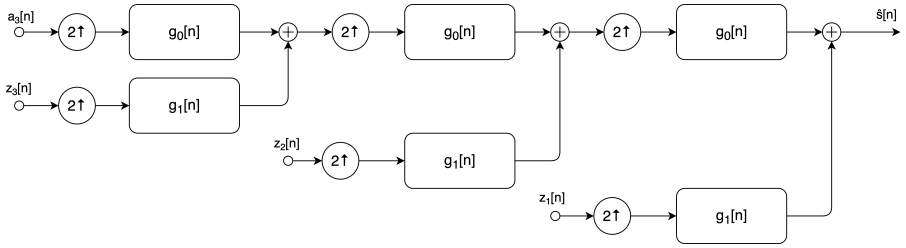


Figure 3.5: Level 3 wavelet transform reconstruction.

3.2 Feedforward Neural Network

It is easy to understand that the motor's velocity is going to influence the noise profile. Smaller velocities lead to smaller noise magnitudes that can cause less interference with other signals while higher velocities produce noise magnitudes that can even mask the whole wanted signals. Also, the conjugation of different motors speeds in the 4 motors will lead to numerous noise profiles. In [27] the idea of relating the motors' speeds to filter bank coefficients is explored. We want to understand if we can improve the system by using a Feedforward Neural Network that predicts the filter coefficients based on the speeds of each motor.

For the implementation of this section we will only apply the FNN to the filter that obtains the best results in section 4.2.

Our FNN will relate the motors' speeds to the resulting filter's coefficients. To do so, we need to start by defining the training process. We pass our inputs, x, first through a linear layer and then by

a non-linear activation function. The linear layer and the activation function are described in 3.38 and 3.39, respectively.

$$\hat{y} = xA^T + b \tag{3.38}$$

$$LeakyRELU(\hat{y}) = \begin{cases} x & \text{if } x \geq 0 \\ 0,01x & \text{if } x < 0 \end{cases} \tag{3.39}$$

where A and b are the parameters adjusted by the NN and $\hat{y}$ is the output of the NN.
After repeating this process as many times as the number of layers defined, we compute the gradient of the loss of this process. The method chosen was Mean Squared Error Loss which is described in equation 3.40:

$$l_n = (y_n - \hat{y}_n)^2 \tag{3.40}$$

where y_n are the filters' coefficients.
Having the gradient of the loss we apply backpropagation in order to update the gradients and find the weights for each layer. Finally, we apply an optimization model so we can retrieve the appropriate weights. From the optimization models available in Pytorch we chose the most used one, Adam [36].

Repeating the described process several times leads to an optimization of the prediction made by the Neural Network. There are still some parameters not defined such as the number of epochs, optimal number of hidden layers and the dimension of each layer. In chapter 4 we will present a small study on which are the three best parameters to use by varying the number of hidden layers between $\{1, 2, 3\}$ and the hidden layer's dimension between $\{2, 4, 6, 8, 10, 16, 20\}$ and see the performance through the number of epochs setting the maximum limit to 5000.

3.3 Neural Network for Speech Recognition

The last method we want to explore to perform Speech Recognition in a UAV is to use only a Neural Network implemented for Speech Recognition that is trained with noisy inputs. We want to understand if the best approach to solve this thesis problem is to denoise the inputs and feed them to a NN to perform the speech recognition or if it's better to teach the NN to recognize noisy inputs through training with data that is already the combination of clean speech and noise from the drones' motors.
In order to quickly test this approach, we are using an already existent Neural Network [37]. When trying to decide which NN to use, we looked for the state-of-the-art NN that have been used for Speech Recognition.
Some words are more likely to appear after a specific word. The same happens with letters individually.

Recurrent Neural Networks [38] are the best type of NN that can deal with this temporal dependency. More specifically, Bidirectional Recurrent Neural Networks allow us to obtain an output based on information from data on the past and the future.
Connectionist Temporal Classification (CTC) is a loss function that has been highly used for speech recognition. For ASR, most of the inputs are observations of a sentence and the outputs are the transcription of the sentence but our inputs might not be in the same number as our outputs. An input might be a single word or part of a word (a phoneme). CTC allows the prediction of the most probable label for a given sequence of inputs. More on Recurrent Neural Networks using CTC loss function can be read in [39].

3.4 Hardware, Software and Datasets

In this section we will mention the hardware and software used in this project and we will also explain how the datasets for testing were created.

3.4.1 Crazyflie

The starting point of this project is based on the work already developed by a researcher at Instituto Superior Técnico (IST). His project consisted on integrating a microphone in a Crazyflie [40]. The Crazyflie is a quadcopter developed by the company Bitcraze that consists on a printed circuit board with 4 motors attached to it. We can see and example of the drone used in this project in figure 3.6.

Figure 3.6: Crazyflie 2.0.

Our first approach was to use the implementation done in the crazyflie to capture the speech and noise signals and add a speech recognition module to the system. The code to implement the ASR

was written in Python due to the necessity of using the program Robot Operating System (ROS) to establish the connection between the computer and the drone. Since ROS only recognizes Python or C++ languages we opted for the first one.

The starting point was developing a code that receives the sound packets captured by the microphone creating an audio file with them. This audio file is then sent to Google Cloud's Speech-to-Text and the Application Programming Interface (API) returns in text what was understood. In order to make the program more user friendly, the program calls itself every 20 seconds so that it only shuts down when the user doesn't want to use it anymore.

In order to test the performance of the system, a list of commands was created. These commands can be found in Appendix A. Since the starting point of the project is to use the drone indoors helping the elderly performing tasks, the commands are all related to activities that can be done at home. Still, it is important to mention that the system can recognize other types of speech and commands that are not present on this list.

Early in the development of the project we understood that the quality of the recordings was very poor which was leading to bad results. The cause of these results was not the performance of the system but the bad quality of the dataset. The DREGON dataset was then used to train and test our system.

3.4.2 DREGON Dataset

The DREGON dataset [41] is a dataset created with the purpose of pushing research in sound source localization using a microphone array embedded in an UAV.

The DREGON dataset was created using the MikroKopter quadrotor UAV and an 8-channel microphone array. An image of the drone used can be seen in figure 3.7.

Figure 3.7: MikroKopter quadrotor with 8-channel microphone array.

The great advantage of using this freely available dataset is that it is composed of recordings of the

motors' noise which are accompanied by information on the motors' speeds at the time of recording. Also, the most valuable data provided for our project is the recordings of each motor individually at a fix speed.

For purpose of testing we started by recording the commands referred in Appendix A with the microphone of a mobile phone in an environment with minimum background noise. To create the noisy speech inputs we opted to only use the files provided with the recordings of each motor individually at the speeds of $\{50, 60, 70, 80, 90\}$ rotations/second. Since we are developing a system to work with a mono-channel microphone we began by creating separate files for each channel. Then, we randomly combined the audio files so we could get different different speed combinations that represent different kinds of movement of the drone (hoovering, going up, moving to the right, etc). The last set created was the mixtures of the noise-only files with the previously recorded commands. The process of mixing the files was repeated for 5 different SNR values, $\{-10dB, -5dB, 0dB, 5dB, 10dB\}$ according to equation 3.41.

$$SNR = 10log_{10}\Big(\frac{P_{speech}}{P_{noise}}\Big) \tag{3.41}$$

3.4.3 LibriSpeech ASR Corpus

To train the RNN we opted to use an already existent and widely used dataset, the LibriSpeech ASR Corpus [42]. For this project we randomly chose parts of the 100 hours set of clean speech and mixed these audio files with the noise-only files previously created.

3.4.4 Software

The software used in this project was, on a first stage, Python and ROS for the reasons previously mentioned. When changing for the DREGON dataset, all the audio files were processed using the program Audacity. From that point forward we used the program Matlab to implement and apply our filters to the noisy inputs and we opted to use Python to perform speech recognition and results' classification. Still in the Python language we chose Pythorch to deal with NN.

3.5 Classification for Testing

WER is the chosen metric to evaluate the performance of the systems. WER is computed according to the equation 3.42

$$WER = \frac{S + D + I}{N} \tag{3.42}$$

where S is the number of substitutions, D the number of deletions and I the number of insertions in the hypothesis and N is the total number of words in the reference. These computations are made using a code from [43] where the inputs are the text file with the commands in Appendix A as the reference and the text files with the output from the Speech Recognizer as the hypothesis.

4

Results

Contents

On this chapter we will start by presenting the test results of the Speech Recognizer when there is no filter applied. After that, we will present the results for five different SNR values when we apply the five filters explained in the previous chapter. To finish the analysis of the filters we will test the performance of the SR system when we add a Neural Networks to the computation of one filter's parameters. At last we will test the performance of the Neural Network created to recognize speech with noise. The code developed for the filtering algorithms is available in [44].

4.1 Speech Recognition without Filtering

To begin the project we want to know what is the influence of the noise produced by the drone in the performance of a speech recognition system. To start the implementation we tested a series of already existent SR systems offered by several companies. We quickly understood that Google Cloud's Cloud Speech-to-Text provided the most reliable results. The decision of using an already existing speech recognizer instead of creating one was made due to the fact that Google has access to a training dataset of a size that we would never be able to match and also has computational resources to train a neural network that leads to greater results than what we could obtain if we created the NN. With this in mind, we opted to focus only in the creation of the noise filtering to create better data to feed this very reliable system to perform the speech recognition.

4.1.1 Crazyflie

To analyze the decrease of performance of the Speech Recognizer we started by recording the commands with the microphone implemented in the drone while the engines were not working. The value obtained for the WER in this case was $39,196\%$.
At this stage of the experiments we had already realized that the integrated microphone in the drone was providing signals with very poor quality so, with the aim of simplifying the tests, we recorded the sound made by the drone while flying and without any speech involved using a recorder and attaching a small microphone with a wire to the drone. Later, with the help of the program Audacity [28], we overlapped the recorded track of the noise with the recorded tracks of the commands. Google's API was never able to formulate a hypothesis and this was clear when we listen to the track. The sound of the noise is so loud that completely mutes the speech. This is translated in a WER of 100%.

4.1.2 DREGON dataset

For the DREGON dataset we started by computing the WER for the speech commands without noise recorded with the cellphone microphone. For this case we obtained a WER of $3,046\%$ This result is as expected due not only to the conditions of recording mentioned in the previous chapter but also due to the fact that the cellphone is composed by two microphones in order to perform noise cancellation.

The last test performed to the dataset without filtering is when we added the noise-only to the speech-only audio files. This time we obtained a WER of $17,172\%$. The first remark to do is that the quality of the microphones, both the ones that recorded the motors' noise and the speech is much better than in the case of Crazyflie. The next remark is that the SNR depends on the speeds of the motors. In Appendix B, table B.1 shows the values of the WER obtained individually for each command and the corresponding SNR. Actually we cannot find a linear dependency between the obtained WER and the corresponding SNR but what we can affirm is that almost all the commands have a positive SNR or the negative ones are at least very close to $0dB$. We believe this fact has a great impact on the satisfying outcome of the ASR system without any kind of filtering. To show the influence of the SNR on the final results we will force the SNR values to be $-10dB$, $-5dB$, $0dB$, $5dB$ and $10dB$ by changing the energy of the speech-only signal before mixing it with the noise-only signal. Table 4.1 shows the WER values obtained for the different SNR.

Table 4.1: WER for no filtering with DREGON dataset.

	$SNR = -10dB$	$SNR = -5dB$	$SNR = 0dB$	$SNR = 5dB$	$SNR = 10dB$
WER	94,949%	75,253%	53,030%	45,455%	30,808%

As it is clear now, the values of WER decrease when we increase the SNR. Although the outcomes tend to be more acceptable when the energy of speech is greater, we want to reduce this percentage to as close to 0 as possible. The $30,808\%$ doesn't seem as bad as compared to the other percentages obtained but it is still 61 out of 198 words that were wrongly guessed.

In the next section we will try to improve the WER by applying different filtering techniques.

4.2 Speech Recognition with Filtering

The results of the previous section support the purpose of this thesis. We will now present the results obtained when the filters explained in the previous chapter are applied to the sound files. From now on, the DREGON dataset was the only dataset used. This decision was made since the quality of the recorded files with the microphone in the Crazyflie is very low which led to a bad performance of the SR system not due to the bad performance of the filters but because of the overall low quality of the sound

files.

4.2.1 Wiener Filter

The first filter tested was Wiener Filter. For the correct usage of the implementation in Matlab, we need to choose as input a segment of the audio file where every motor is working at the same time. Also, the period of $2s$ in the beginning of the segments was chosen as the period of no speech (noise-only) to compute the noise statistics. The WERs obtained for the first filter are documented in table 4.2.

Table 4.2: WER for filtering with Wiener Filter.

	$SNR = -10dB$	$SNR = -5dB$	$SNR = 0dB$	$SNR = 5dB$	$SNR = 10dB$
WER	99,492%	96,954%	89,848%	74,619%	59,898%

As it was expected we can see a decrease of the WER when the SNR increases. Since the Wiener Filter is such a popular filter used for noise removal, we were expecting better results. After looking at the numbers, we listened to the audio files. Actually what happens is that there is a clear noise removal (in the absence of speech period there is almost no noise) but the voice is distorted and unclear. This distortion decreases as we increase the speech energy and that explains the improvement of results with the increase of the SNR values. This filter is indeed able to have a good performance in noise-only segments but it is not able to achieve a satisfactory reconstruction of the speech-only signal.

4.2.2 Least-Mean-Square Adaptive Filter

For the testing of the LMS Adaptive Filter we changed the parameter μ and the order of the filter. The results obtained for the different values of SNR are explicit in tables 4.3 to 4.7.

Table 4.3: WER for $SNR = -10dB$ for LMS Adaptive Filter.

WER	$\mu = 0,01$	$\mu = 0,05$	$\mu = 0,075$	$\mu = 0,1$	$\mu = 0,5$
order = 2	6,599%	5,051%	5,051%	4,545%	12,690%
order = 5	5,076%	4,040%	5,556%	5,556%	38,579%
order = 7	6,061%	5,556%	5,556%	5,556%	52,792%
order = 12	7,107%	5,076%	7,071%	7,071%	70,051%
order = 20	5,076%	5,051%	8,081%	11,616%	89,899%

Based on the values of WER obtained we will set the two parameters to be used from now on. On table 4.3 we find the lowest WER value for a 5^{th} order filter with $\mu = 0,05$. Since we have in total 198 words, $4,040\%$ corresponds to 8 words that were wrongly guessed. On table 4.4 the best

Table 4.4: WER for $SNR = -5dB$ for LMS Adaptive Filter.

WER	$\mu = 0,01$	$\mu = 0,05$	$\mu = 0,075$	$\mu = 0,1$	$\mu = 0,5$
order = 2	4,040%	3,535%	4,040%	3,535%	13,706%
order = 5	5,051%	3,535%	4,040%	6,061%	35,533%
order = 7	4,040%	4,545%	5,051%	4,545%	50,505%
order = 12	5,584%	4,569%	5,556%	7,071%	67,005%
order = 20	6,599%	6,061%	6,566%	10,101%	89,394%

Table 4.5: WER for $SNR = 0dB$ for LMS Adaptive Filter.

WER	$\mu = 0,01$	$\mu = 0,05$	$\mu = 0,075$	$\mu = 0,1$	$\mu = 0,5$
order = 2	3,030%	3,535%	3,030%	4,545%	11,168%
order = 5	3,046%	3,030%	6,566%	8,081%	31,980%
order = 7	4,040%	4,545%	5,556%	7,071%	42,929%
order = 12	4,061%	4,061%	6,061%	6,061%	61,929%
order = 20	4,061%	5,556%	9,091%	8,081%	82,323%

Table 4.6: WER for $SNR = 5dB$ for LMS Adaptive Filter.

WER	$\mu = 0,01$	$\mu = 0,05$	$\mu = 0,075$	$\mu = 0,1$	$\mu = 0,5$
order = 2	2,538%	3,553%	4,569%	5,076%	14,213%
order = 5	1,523%	5,584%	5,556%	6,566%	32,487%
order = 7	4,040%	4,545%	6,566%	5,584%	41,919%
order = 12	4,569%	5,076%	6,091%	7,576%	59,898%
order = 20	4,569%	7,071%	7,576%	10,101%	75,758%

Table 4.7: WER for $SNR = 10dB$ for LMS Adaptive Filter.

WER	$\mu = 0,01$	$\mu = 0,05$	$\mu = 0,075$	$\mu = 0,1$	$\mu = 0,5$
order = 2	6,599%	5,076%	5,076%	6,599%	15,657%
order = 5	4,569%	4,061%	6,599%	7,614%	40,909%
order = 7	4,545%	4,569%	7,576%	8,586%	48,485%
order = 12	3,553%	5,076%	6,599%	7,107%	61,421%
order = 20	3,553%	7,107%	10,152%	13,131%	72,727%

results obtained were 7 wrong words ($WER = 3,535\%$) for the parameters $\{order = 2, \mu = 0.05\}$, $\{order = 2, \mu = 0,1\}$ and $\{order = 5, \mu = 0,05\}$. As for table 4.5 the best performance was obtained for the sets $\{order = 2, \mu = 0.01\}$, $\{order = 2, \mu = 0.075\}$ and $\{order = 5, \mu = 0.05\}$ with a WER of $3,030\%$ (6 incorrect words). It is clear that for the cases when the energy of the noise is greater or equal to the energy of speech the parameters that optimize the ASR are the filter order = 5 and $\mu = 0,05$.

When we look to tables 4.6 and 4.7 we realize that these parameters are not the optimal when the

SNR is positive. In table 4.6 we obtain the best WER percentage for $\{order = 5, \mu = 0.01\}$ with only 3 incorrect words (the closest value obtained to the wanted no error). In table 4.7 the best result obtained is $3,553\%$ for the sets $\{order = 12, \mu = 0.01\}$ and $\{order = 20, \mu = 0.01\}$.
Neither of these 3 sets mentioned before correspond to the best performances when $SNR \leq 0$. Since we are more concerned about improving the performance of the ASR system in the cases where SNR is worst, we will evaluate the impact of using the parameters $\{order = 5, \mu = 0.05\}$ for $SNR = 5dB$ and $SNR = 10dB$. In the first case we have an error of $5,584\%$ which corresponds to 11 wrong words. Our best performance was mentioned to be 3 incorrectly guessed words. It is true that in this situation the usage of these non-optimal parameters has a significant impact on the performance of the ASR system but considering absolute values, guessing 11 words incorrectly out of 198 words is still an error inferior to 10% and acceptable. It becomes even more acceptable when we look at the guesses made by the ASR system. We realized that some of these incorrect guesses are because the word was a singular noun and the ASR system guessed it as a plural noun or mistakes between the word "to" and "the" which are errors that might be overcome when in Natural Language Processing usage. For the second case we get 8 wrong words $(4,061\%)$ instead of the 6 wrong words obtained by the best fit parameters. With a difference of only 2 more incorrect words we believe that this increase of error is compensated by the decrease of computations involved in finding the 5 filter coefficients instead of the 12 or 20. With all of these considerations from now on we will use the set of parameters $\{order = 5, \mu = 0.05\}$ and have in mind the results in table 4.8.

Table 4.8: WER for filtering with LMS Adaptive Filter.

	$SNR = -10dB$	$SNR = -5dB$	$SNR = 0dB$	$SNR = 5dB$	$SNR = 10dB$
WER	4,040%	3,535%	3,030%	5,584%	4,061%

In opposition to the previous filter, this time we can affirm that the performance of the ASR system does not depend on the SNR values. Also, the WER values obtained are very close the the ones obtained when there was no noise $(3,046\%)$. When listening to the audio files, in the beginning we can still listen to the noise of the motors but it decreases with time until the point that no sound is listened. When the segment of the speech command arrives we listen to it with almost no distortion making its perception very clear. Indeed we verify that the speech commands are as clear when $SNR = -10dB$ as for the case when $SNR = 10dB$. Although the outcome using this type of filtering was very satisfactory and extremely suitable in solving this project's problem there is a major limitation. It is necessary to have the noise-only profile in order for the filter to work. This problem will be addressed further in the document.

4.2.3 Noise Gate

To test the performance of this filter we tested the sound files not only for the different SNR values, as it was done for the other filters, but also for three different sets of thresholds. On the equation 3.23, we applied the values 1, 1,5 and 2 to n. Table 4.9 show the obtained WER for different values of SNR and standard deviations used in the algorithm.

Table 4.9: WER for filtering with Noise Gate with different deviations.

WER	$SNR = -10dB$	$SNR = -5dB$	$SNR = 0dB$	$SNR = 5dB$	$SNR = 10dB$
$\sigma = 1$	96,954%	89,848%	76,142%	59,391%	43,655%
$\sigma = 1,5$	98,985%	95,431%	80,203%	69,036%	53,807%
$\sigma = 2$	99,492%	95,431%	84,772%	69,036%	59,391%

We would guess that a standard deviation of $1,5$ would show better results because it would take into consideration the cases when the noise has a greater difference to the average noise computed but what actually happens is that increasing the weight of the deviation leads to the consideration of speech segments as noise. From now on we will therefore use the values corresponding to $\sigma = 1$ that are present in table 4.10.

Table 4.10: WER for filtering with Noise Gate.

WER	$SNR = -10dB$	$SNR = -5dB$	$SNR = 0dB$	$SNR = 5dB$	$SNR = 10dB$
	96,954%	89,848%	76,142%	59,391%	43,655%

Once again we detect the relation mentioned before between the WER and the SNR values. Listening to the audio files we can state that until now this is the algorithm that performs less noise removal. What we actually listen to is the constant presence of the motors' noise that is increasingly more attenuated as the SNR values increase. This attenuation is the origin of the speech enhancement that we are able to detect. This goes according to the explanation given in section 3.1.3 since we only decrease the noise and we maintain the speech. There is almost no distortion in the listened commands because we apply the filter not only to the magnitude of the signal but also to its phase but, as it is clear from the results, there is still a lot of interference by the motors' noise.

4.2.4 Motion-Dependent Spectral Subtraction

For the Motion-Dependent Spectral Subtraction (MDSS) filter we set the values of α and β to 2 and $0,5$, respectively. Table 4.11 shows the WER results obtained when we apply the algorithm to the noisy audio files.

Table 4.11: WER for filtering with Motion Dependent Spectral Subtraction.

WER	$SNR = -10dB$	$SNR = -5dB$	$SNR = 0dB$	$SNR = 5dB$	$SNR = 10dB$
	93,939%	77,273%	52,525%	43,939%	30,808%

The presence of the strong correlation between the results obtained and the SNR values is very clear also for this type of filtering. Actually, when listening to the resulting audio files we notice only a very small attenuation of the sound (speech and noise) and, sometimes, the noise is even harsher than in the not filtered file. Also, since we consider the noise as a constant value within each channel, we actually introduce more noise in the silent parts.
In addition to the not satisfactory results, we also noticed that this algorithm had a greater computational cost than all the others and that the running time was also not acceptable since we want to integrate this filter in a real time system.

4.2.5 Discrete Wavelet Transform

For this last filter there were two decisions that had to be done in order to implement the algorithm. The first one was to decide how many levels of decomposition we would use. The second one has already been mentioned on section 3.1.5, which type of thresholding will be used.
Concerning to the first problem, we chose 17 levels of decomposition. This choice was made based on listening the outcomes and computing the error between the speech-only signal and the predicted speech signal. What we noticed was that the error was smaller for levels of decomposition between 14 and 17 but when we listened to the resulting audio files, the voice presented distortions. For a decomposition in 17 levels the voice suffered less distortion and the outcome was similar to greater levels of decomposition but with less computational cost.
As for choosing the best threshold method for noise removal we preformed the algorithm with the 4 different types of threshold to the command number 2 for a $SNR = 10dB$. In figure 4.1 we can observe the difference between the predicted speech signal and the clean speech signal. It is clear that using Universal Threshold we have the lowest error. The computation of the threshold is indicated in equation 4.1.

$$\lambda = \sqrt{2ln(N)} \tag{4.1}$$

where λ is the threshold and N is the number of samples of the noisy signal.

Once all the parameters were defined we computed the WER for the 5 cases that we have been evaluating and the results obtained are presented in table 4.12.

Looking at the numbers it is clear that the algorithm doesn't perform good, specially for the cases

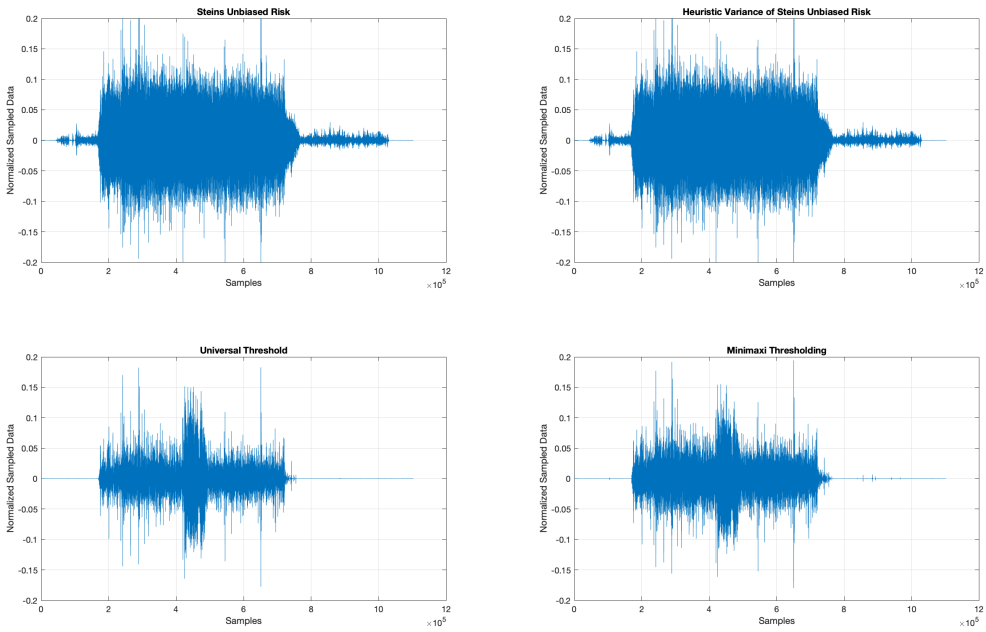


Figure 4.1: Effect of different methods of thresholding in noise removal.

Table 4.12: WER for filtering Discrete Wavelet Transform.

WER	$SNR = -10dB$	$SNR = -5dB$	$SNR = 0dB$	$SNR = 5dB$	$SNR = 10dB$
	99,495%	94,949%	82,323%	60,606%	49,495%

where the noise is higher than the speech. This outcome was expected since this algorithm is mainly used to remove noise encountered in signals with much less oscillations than in our case. The audio files generated by this algorithm still present lots of noise. In the cases of negative SNR the noise removal applied is still not enough to enhance the speech signal and for the other cases we notice that the noise is almost the same but the speech signal is enhanced. We also notice that with the increase of the SNR, the predicted speech suffers more distortion. This filter doesn't show a promising solution for our problem.

4.2.6 Conclusions

Based on the results obtained in the previous subsections, we elaborated the graphics in figures 4.2 to 4.7.

In the first five figures we are able to analyse and compare the influence that each of the previously explained filters have on the performance of the ASR system. In the color graphs we can find the WER values for the case when no filtering is performed on the audio signal (Original), highlighted in orange,

and the cases when Wiener Filter (WF), Least-Mean Square Adaptive Filter (LMS), Noise Gate (NG), Motion Dependent Spectral Subtraction (MDSS) and Discrete Wavelet Transform (DWT) are applied to the noisy signal for speech enhancement. Figure 4.7 shows the performance of all the cases in analysis along the SNR values.

In all figures we can see that 3 of the filters have worsen the results. Although Wiener Filter, Discrete Wavelet Transform and Noise Gate have removed some of the noise in the original audio file (and that is clear when we listen to them), they also ended up introducing more distortion to the speech segments leading to the bad outcomes of the ASR system.

Specially in figure 4.7, we can see that the results for the MDSS algorithm are extremely close to the no filtering results. This shows that the statement done in section 4.2.4 about the attenuation of noise being very soft is indeed true and that the computational cost associated to the denoise of the signal is not worth.

In every figure it is clear that the LMS Adaptive Filter introduces a great improvement to the systems results and has an almost perfect performance. This happens due to the constant adaptation through time to the noise of the motors and also due to the fact that the noise is uncorrelated to speech. This uncorrelation makes it possible to isolate the speech signal with no distortion leading to WER values in the neighborhood of the WER obtained before introducing noise ($3,046\%$). We conclude that apart from the LMS Adaptive Filter, none of the other filters are apt to improve our results. In fact, most of the times they decrease the performance of the system increasing its computational costs.

On the other hand, the LMS Adaptive Filter proved to be a great solution for our problem and its usage will be further developed in the next section.

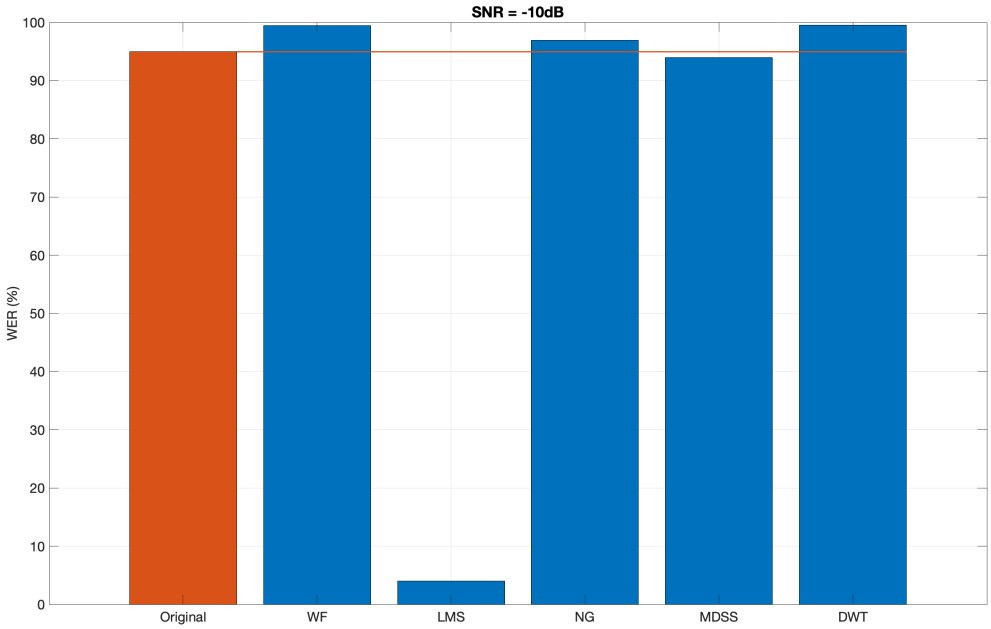


Figure 4.2: WER for original file and all the filters applied when SNR=-10dB.

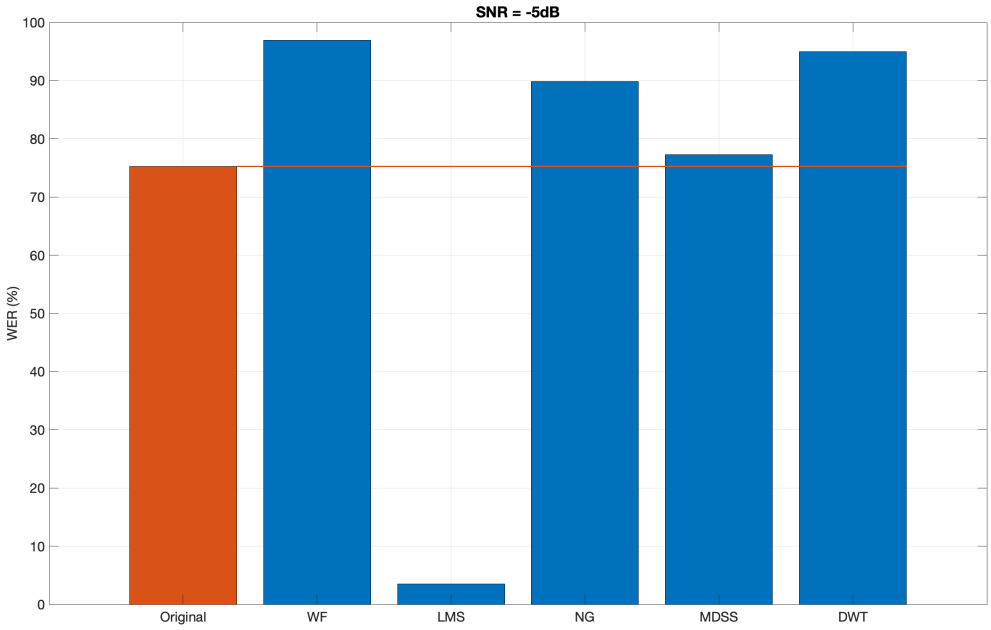


Figure 4.3: WER for original file and all the filters applied when SNR=-5dB.

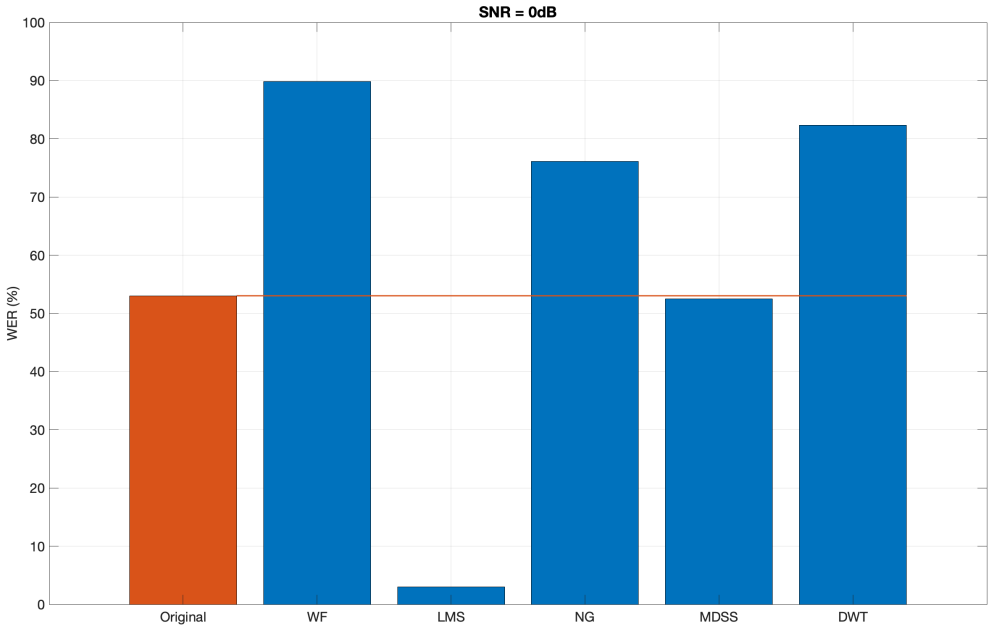


Figure 4.4: WER for original file and all the filters applied when SNR=0dB.

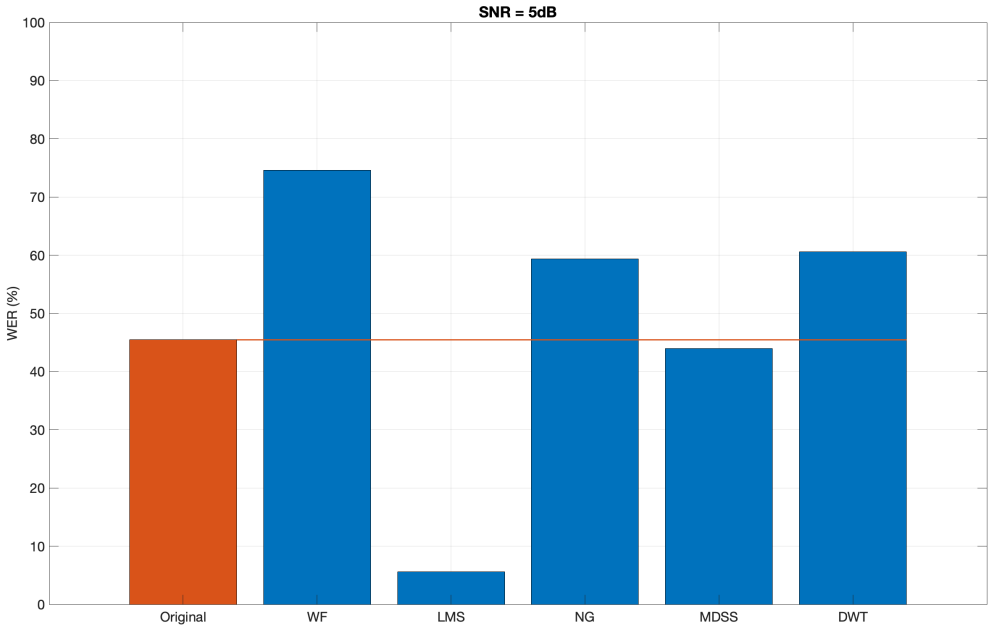


Figure 4.5: WER for original file and all the filters applied when SNR=5dB.

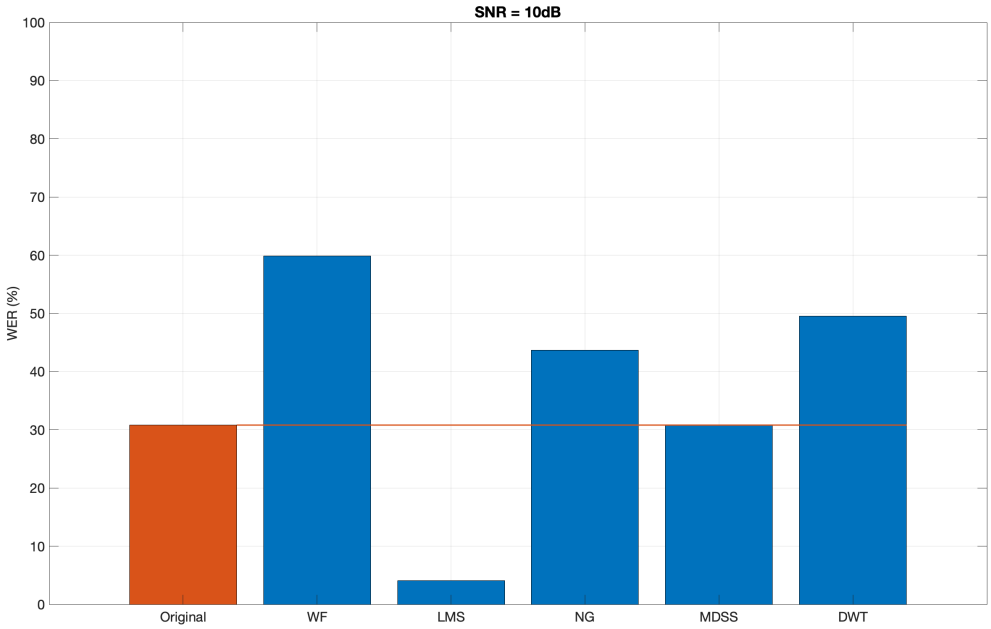


Figure 4.6: WER for original file and all the filters applied when SNR=10dB.

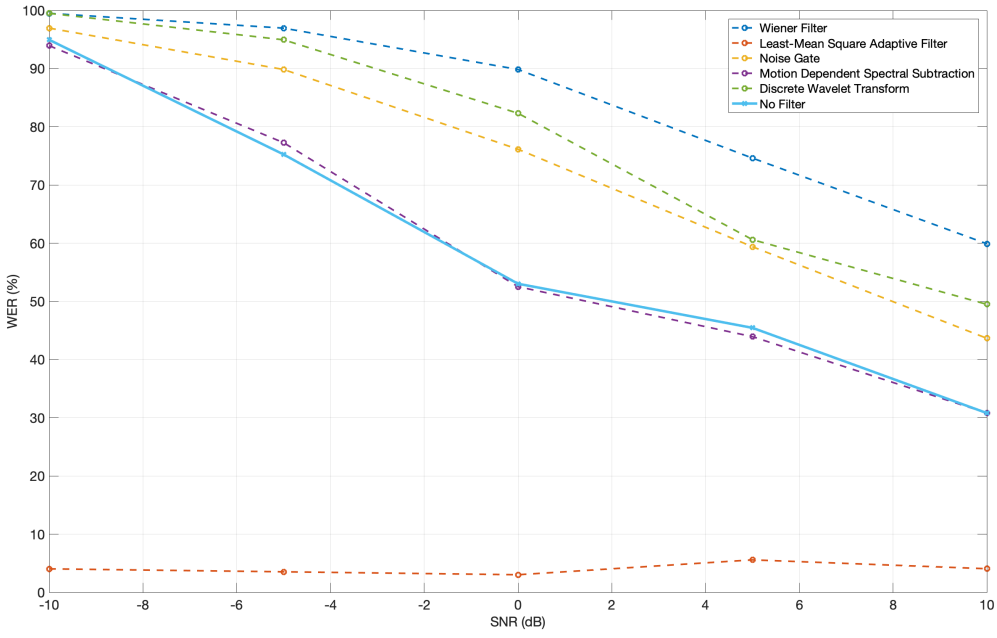


Figure 4.7: Comparison of results between original signal and all filtered signals for all the SNR values.

4.3 Speech Recognition with Filtering using Neural Networks

From the results obtained in section 4.2 we will now try to improve our speech recognizer system by relating the LMS Adaptive Filter to the known velocities of each motor.

4.3.1 Applying the Neural Network to the LMS Adaptive Filter

The starting point of this section is to define the parameters of the Neural Network. From each of the training files we have the velocity of each motor (that is constant through the file) so the input of our NN is of size 4. As for the output, we found that the best performance is when we have a 5^{th} order filter so the outputs are the coefficients computed through equation 3.17, $\boldsymbol{w}[n]$, of dimension 6.

In order to find the best parameters we need to set a classifier. For our case we decided to use again the MSE.

$$cost = \sum_{n=1}^{N} l_n \tag{4.2}$$

In equation 4.2, l_n is as defined in equation 3.40 and N is the total number of outputs for the training set. Since our training set is composed by 266 files and each file produces 6 outputs, $N = 1596$.

Figure 4.8 shows the three plots corresponding to the number of hidden layers (hl) used. These plots show the cost associated to the number of epochs when we have different hidden layers' dimensions.

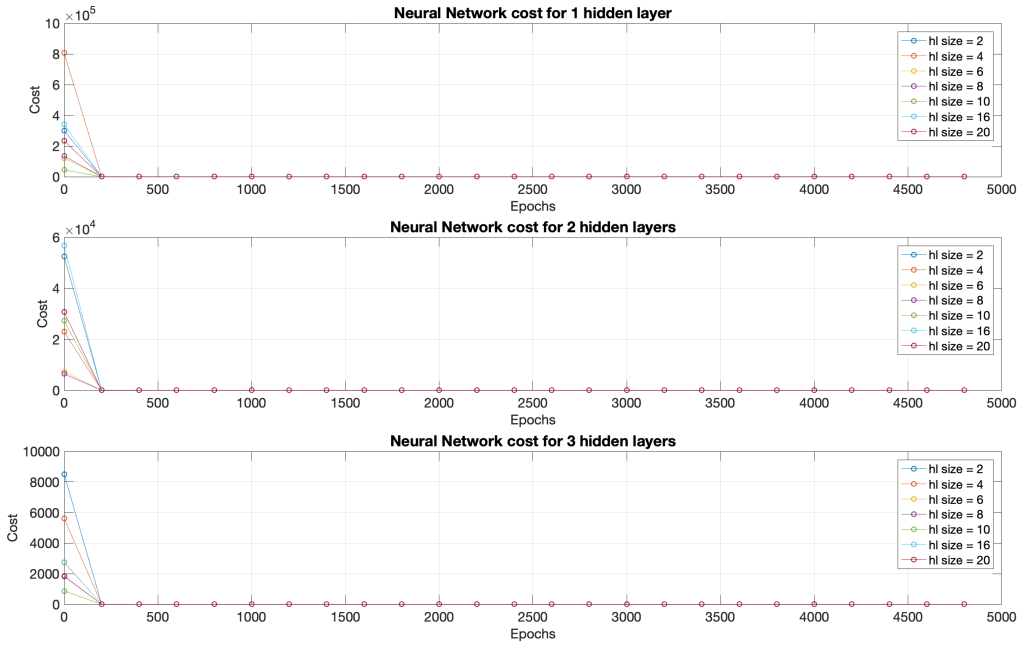


Figure 4.8: Learning curve for different number of hidden layers and hidden layers' dimensions.

Figure 4.8 isn't optimal to take good conclusions for the definition of our NN. The only information we can retrieve is that in less than 500 epochs the NN converges to the best predictions. We will now zoom in the graphics to take better conclusions.

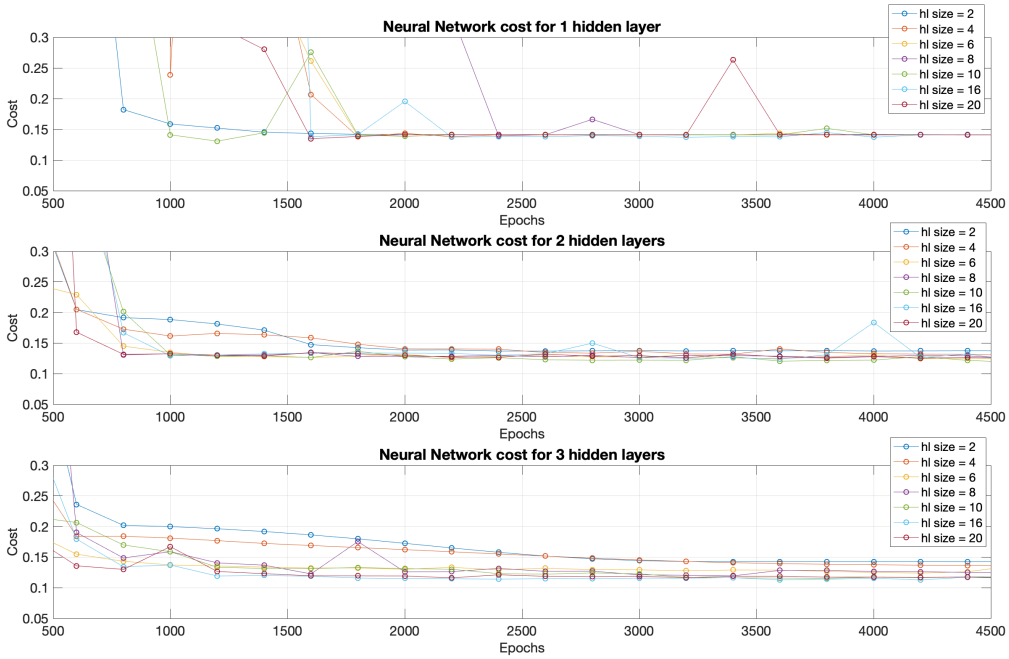


Figure 4.9: Zoom of the earning curve for different number of hidden layers and hidden layers' dimensions.

What we can actually observe in figure 4.9 is that when we increase the number of hidden layers, the number of epochs necessary to converge for the best result is smaller. Also, from this figure we cannot define the best number of elements for the hidden layer so we present table 4.13 that shows the cost after stabilizing associated to each NN when we change the two parameters we've been analyzing.

Table 4.13: Cost associated to the NN when there's change of the number of hidden layer and its dimensions.

Cost	$1\ hidden\ layer$	$2\ hidden\ layers$	$3\ hidden\ layers$
$hl\ size = 2$	0,1412	0,1373	0,1423
$hl\ size = 4$	0,1409	0,1324	0,1356
$hl\ size = 6$	0,1410	0,1241	0,1298
$hl\ size = 8$	0,1410	0,1252	0,1231
$hl\ size = 10$	0,1410	0,1191	0,1181
$hl\ size = 16$	0,1418	0,1220	0,1131
$hl\ size = 20$	0,1413	0,1222	0,1162

As we can see, the values are not very different. From the worst case to the best case we have a difference of $0,0287$. From table 4.13 we see a slight improve of the cost when we increase the number

of hidden layers. Since we want to have a system that gives the best results with less computational cost, we opt to implement the NN with 2 hidden layers with size 10. Having in mind figure 4.9 it is true that the number of epochs necessary to achieve the optimal results is greater but we believe this occurrence is compensated by the decrease of complexity of the system and by the outcome.

Having these parameters settled we will now present in table 4.14 the performance of the ASR system when we directly apply the coefficients computed through the NN to the filtering process.

Table 4.14: WER for filtering with LMS Adaptive Filter using a Neural Network.

WER	$SNR = -10dB$	$SNR = -5dB$	$SNR = 0dB$	$SNR = 5dB$	$SNR = 10dB$
	3,046%	3,553%	3,046%	4,061%	3,553%

The results obtained are still very close to the optimal outcome (0%). We can now conclude that it is possible to use a NN to relate the motors' speeds information to the needed filter coefficients. Let's now compare these results with the ones obtained without using the Neural Network.

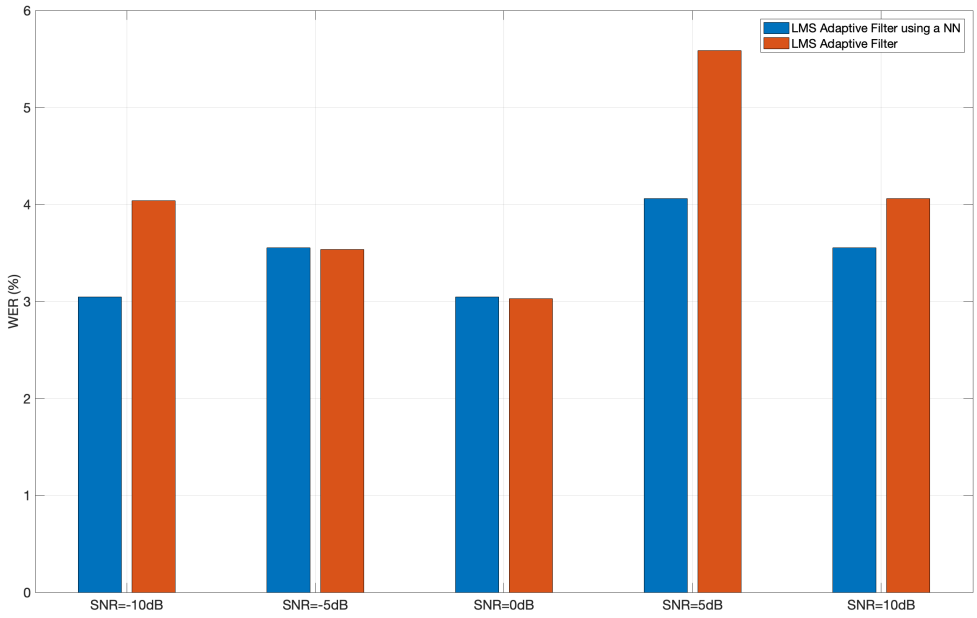


Figure 4.10: Comparison between results from the LMS Adaptive Filter with and without prediction from Neural Network.

In figure 4.10 we can see that the usage of the Neural Network ended up improving the system's results. It would be expected that the WER would increase since the Neural Network presents a small estimation error. Listening to the resulting files what we hear now is a constant through the whole file but almost inaudible noise. Another evidence is less distortion in the speech signal for some cases. This

is the reason why we believe the results have improved. Still, we think that it's important to mention that the maximum difference of wrongly guessed words is of $1,523\%$ (3 words) and when we analyzed the outputs of the ASR system some of the words were constantly guessed wrongly independently of the SNR.

Relating the velocities of each motor of the drone to the filter's coefficients is possible and might improve the prediction of the ASR system. Although it is not shown in this document, we tried to find a model that described this relation. It was clear from the beginning that we could only obtain the coefficients of the filter through the usage of a Neural Network.

4.3.2 Real life adaptation of the system

Although the previous results look very promising, there is a big issue regarding to the implementation of the algorithm in a drone. We are only able to perform the denoising if the noise-only profile perfectly matches the noise profile of the noisy signal. What happens in real life is that we cannot capture the noise-only profile of the drone. Specially in a small UAV the integrated microphones will always capture both the noise and the speech due to the drone's size. To overcome this problem we came up with the idea of capturing the noisy signal at different SNR by using two directive microphones with one facing the motors and the other facing the opposite side or by integrating two microphones: one that is very close to the motors and will be used in the algorithm as if it was the noise-only signal and another one that will have a greater SNR because it will be implemented further away from the motors. Using the coefficients previously predicted by the NN, we will now obtain the clean speech signal through the subtraction of the signal with higher SNR and the filtered signal with lower SNR.
In order to test the performance of this improved system, we will start by analyzing which is the minimum difference of SNR between the signals captured by the two microphones that can be used to obtain acceptable results. To do so, for command number 2 (which was a command always correctly guessed in the previous tests with the LMS Adaptive Filter) we will use the signal with $SNR = -10dB$ as the noise input and we will evaluate the performance of the system when as the noisy signal input we increase the SNR by steps of $0,25dB$. Table 4.15 shows the values of the difference of SNR tested and the WER obtained.

Table 4.15: WER of command 2 for filtering with LMS Adaptive Filter using a Neural Network adapted for real case.

$\Delta SNR(dB)$	0	$0,25$	$0,5$	$0,75$	1	$1,25$	$1,5$	$1,75$	2	$2,25$	$2,5$	$2,75$	3
$WER(\%)$	100	0	0	0	0	0	0	0	0	0	0	0	0

As it is clear from the results obtained, an increase of $0,25dB$ on the SNR values allows us to obtain the wanted results. Let's look at the following equations to see what actually has to happen to the input

signals.

$$SNR(dB) = 10log_{10}\Big(\frac{P_{speech}}{P_{noise}}\Big) \tag{4.3}$$

for $SNR = 0dB$:

$$\frac{P_{speech}}{P_{noise}} = 1 \tag{4.4}$$

and for $SNR = 0,25dB$:

$$\frac{P_{speech}}{P_{noise}} = 1,06 \tag{4.5}$$

In terms of simplicity, if we consider that the power of noise, P_{noise}, doesn't change, this means that the power of the signal with greater SNR has to be $1,06$ greater than the power of the signal with less SNR. We believe this value can be easily met since in the case of using directive microphones facing opposite sides the SNR will be very different and for the case when we have microphones at different distances, by increasing the distance of one of the microphones to the noise source not only we decrease the power of noise captured but most likely we will be increasing the power of speech due to approximation of the microphone to the speech source.

To finish this small study on the adaptation of the filter to a real case we will now test the algorithm when as the noisy speech input we have the commands at SNR of $-10dB$, $-5dB$, $0dB$, $5dB$ and $10dB$, the noise input is the commands with SNR values of $-10,25dB$, $-5,25dB$, $-0,25dB$, $4,75dB$ and $9,75dB$, respectively, and the coefficients of the filter are the ones previously computed by the Neural Network. The results of this improved system are presented in table 4.16 and the comparison with the two implemented systems using this type of filtering is shown in figure 4.11.

Table 4.16: WER for filtering with LMS Adaptive Filter using $\Delta SNR = 0,25$.

WER	$SNR = -10dB$	$SNR = -5dB$	$SNR = 0dB$	$SNR = 5dB$	$SNR = 10dB$
	44,67%	30,964%	8,629%	6,091%	5,076%

Using a difference of $0,25dB$ between the SNR values of the two signals clearly degrades the performance of the ASR system. What we understood from observing the predicted commands was that in most of the cases it would either correctly guess the whole command or it wouldn't guess a single word. Listening to the resulting audio files we understood that sometimes the speech would also be filtered. This happens because the filter is based in the assumption that the noise input only correlates with the

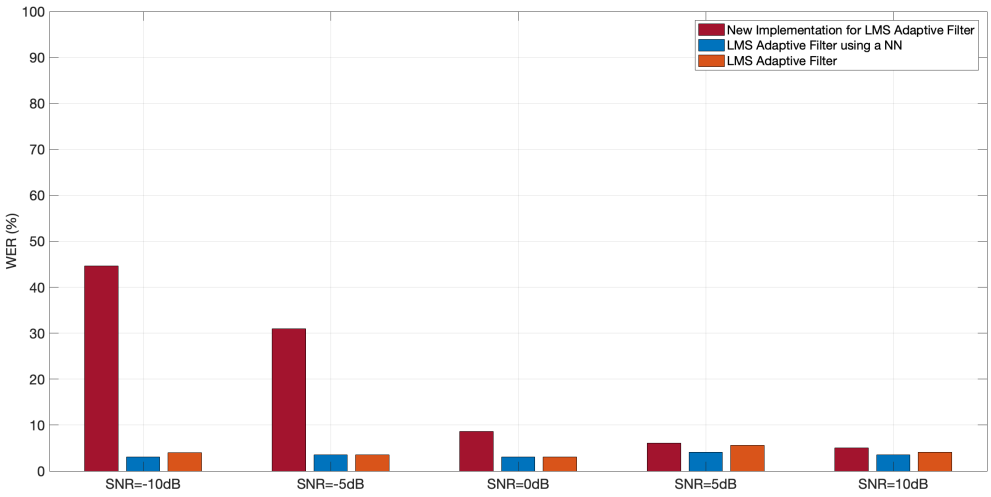


Figure 4.11: Comparison between results from all the systems using LMS Adaptive Filtering.

noisy part of the signal and specially for lower SNR values the speech is masked by the noise and is cut out by the filter. When looking for the best ΔSNR fit we noticed the audio files were more clear when $\Delta SNR = 0,5dB$. According to the logic behind equations 4.3 to 4.5, this would mean that the power of the signal with higher SNR has to be $1,122$ times greater than the other signal which is still a reasonable value. We will now proceed to the same testing using the difference of SNR of $0,5dB$. The results can be seen in table 4.17.

Table 4.17: WER for filtering with LMS Adaptive Filter using $\Delta SNR = 0,5$.

WER	$SNR = -10dB$	$SNR = -5dB$	$SNR = 0dB$	$SNR = 5dB$	$SNR = 10dB$
	14,213%	14,213%	3,046%	2,538%	4,061%

The results from table 4.17 confirmed our motivation to try the experiment with a higher ΔSNR. Figure 4.12 shows once again the comparison between all the implementations with the LMS Adaptive Filter but this time with the new value of ΔSNR.

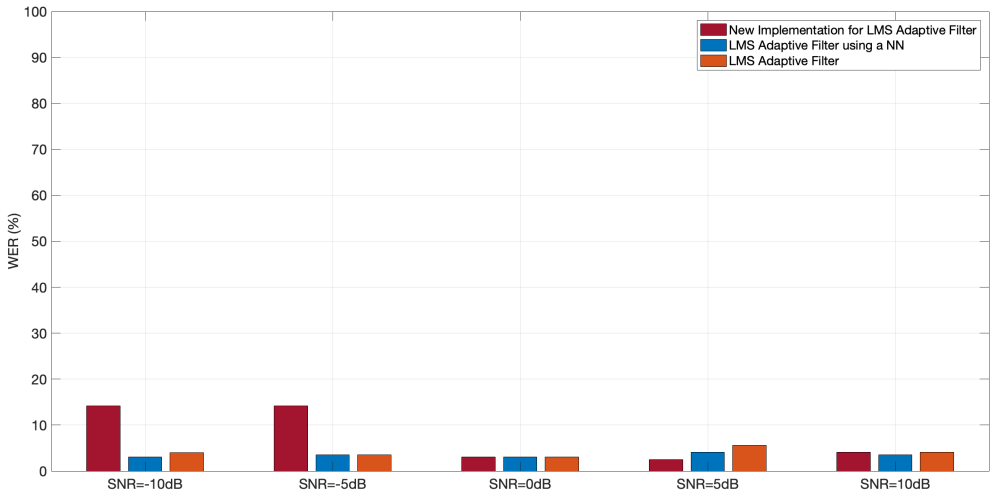


Figure 4.12: New comparison between results from all the systems using LMS Adaptive Filtering.

Figure 4.12 shows that for signals with lower values of SNR the performance of the system is still a little far from the performances previously obtained but for positive values of SNR the system can even perform better than previously. The WER obtained when we have input signals of $SNR = -10dB$ and $SNR = -5dB$ is considerably greater than in the other two cases that we are comparing. These percentages of error are translated in 28 words incorrectly guessed or not guessed at all. The relation we can find from the two sets of testing when varying the ΔSNR is that when we increase the difference between the SNR values of the input signals, we considerably decrease the WER when dealing with signals of negative SNR.

4.3.3 Conclusions

In figure 4.13 we can observe the evolution of results obtained as we improved and adapted the system. On the bars of the left we can see the WER when no filtering was applied to the input signal. On the central bars we have the results obtained when applying the LMS Adaptive Filter with the filtering coefficients being computed as the algorithm performs. The bars on the right show the results when we compute the filtering coefficients through a Feedforward Neural Network and we use as input signals two noisy signals with different values of SNR (for this case $\Delta SNR = 0, 5dB$).

Filtering the input signal, independently of the value of SNR, significantly improved the performance of the system. Although the case of only applying the LMS Adaptive Filter shows an overall best result, as it was already mentioned, it is not realistic to think that we can use this implementation in a real system. The new implementation proposed has a slight worse performance but comparing with the results when there's no filtering, the WER achieved is always better and the values are very satisfying.

This improved system allows us to infer the filtering coefficients from the information on the motors' speeds leading to a straight away denoising and not an adaptive denoising through time. Also, we showed now that it is possible to use the LMS Filter even though we don't have the isolated noise profile. The counterpart of this implementation is the necessity of including another microphone in the UAV but the results presented until now sustain the necessity of doing so.

In conclusion, using a Neural Network to predict filter's coefficients and two microphones to capture input signals at different SNR showed to be a good and relevant solution for the problem stated in this thesis.

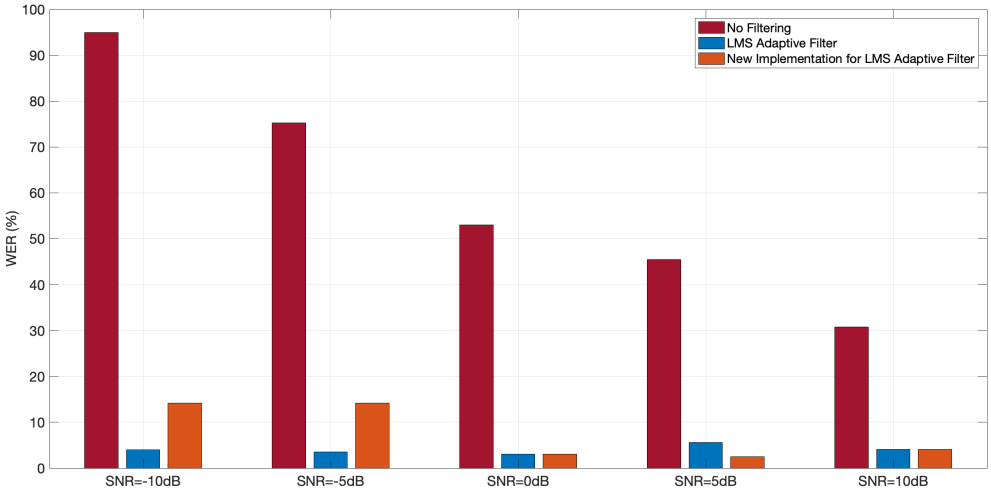


Figure 4.13: WER values for best systems in each section.

4.4 Speech Recognition using Recursive Neural Networks

Based on the work developed in [45] we decided to use the default parameters of the CTC Recurrent Neural Network of 6 hidden layers and change the number of epochs to 100. The number of hidden layers is almost the double as the NN studied in the paper and the number of epochs is within the range studied. Feeding the NN with the LibriSpeech dataset mixed with noise, after approximately two days of running time, we obtained the outputs of the NN for the test set when the SNR is $10dB$ (best scenario of study). The outputs obtained were sequences of letters that wouldn't form words. Furthermore, most of the speech commands were translated into the letters 'o', 'f' and 'r', independently of the phoneme that they are actually made of.

One of the first steps in Recurrent Neural Networks is the feature representation. There is the necessity of converting the audio signal into a representation that can be fed to the NN. The most used representation is MFCC as it has already been explained. In the situation of drones, the noise profile of the motors is highly correlated to their speeds and in lots of cases, this noise profile will mask the speech signals.

Trying to make a correspondence between MFCC and phonemes when there's so much unpredictability associated to the MFCC corresponding to a single phoneme is a task very difficult to overcome. We believe that unless there is a high volume of training data covering all types of noise and phonemes the chances of reaching acceptable results are very low.
Due to the very poor results obtained for the less challenging case of study we decided not to proceed with this approach.

5

Conclusions

During this research we have tested the robustness of various Speech Recognition systems to speech inputs corrupted by noise produced by a drone's motors. In the first set of tests, unfiltered speech signals were fed to the ASR system and the best result of Word Error Rate obtained was $30,808\%$ which is not a value that gives confidence to integrate this type of systems in new technologies. In an attempt to improve the results, 5 filters were applied to the noisy speech inputs so a cleaner speech signal could be fed to the Speech Recognizer. The Wiener Filter, Noise Gate, Motion-Dependent Spectral Subtraction and Discrete Wavelet Transform revealed not to be good contributions to the solution of the problem since they would either distort the speech signal or not attenuate/eliminate the noise as needed. The Least-Mean-Square Adaptive Filter is presented as the right and very promising solution for the denoising problem being the worst WER result obtained of $5,584\%$. After finding the best filter, a Feedforward Neural Network was developed to predict the filter's coefficients based on the information of the speeds of each motor. The fact that the denoise is made in real time and not adaptively enabled the possibility of matching or even improving the previously obtained WERs. In practical terms, this filter could not be applied to a real system since it would be needed the noise-only signal as one of the inputs. A slight alteration to the method is proposed. In the new implementation a signal captured with less SNR is used as the noise-only input and another signal captured with higher SNR is used as the noisy speech. In conjugation with the coefficients available through the training of a NN in ideal conditions (when the noise signal input is indeed noise-only) we were able to generate results that, for better conditions (higher SNR of the input noisy speech), matched the previously tested systems and, for worst conditions (negative SNR input signals), the percentage of WER is not ideal but still acceptable to start introducing this technology in real systems. Lastly, the training of a Recurrent Neural Network for Speech Recognition with speech already corrupted by ego-noise of a drone was attempted but the results showed up not to be competitive with the approach of filtering the noise of speech before feeding it to a Speech Recognizer. The proposed system using LMS Adaptive Filtering and a Neural Network for filter's coefficients prediction is presented in the scheme of Figure 5.1. This system is robust against different values of SNR of the inputs which is the main challenge to overcome in the problem studied.

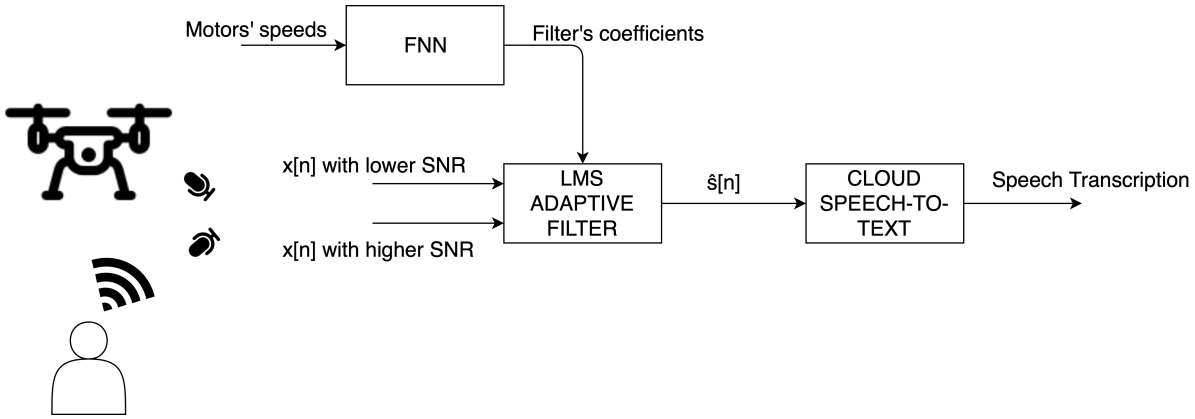


Figure 5.1: Scheme of the proposed ASR System.

There are still lots of opportunities of development on the presented subject. To give continuation to this research it would be important to confirm the results obtained through testing in a real implementation. In the implementation it would be interesting to discover which is the best way of integrating the two microphones capturing the noisy signals with different signal-to-noise ratios. It would also be important to confirm if in a real implementation the theoretical value presented of $\Delta SNR = 0,5dB$ between the two input signals is enough and achievable.

Since NN for Speech Recognition is still an area being widely researched, trying to create end-to-end Neural Networks for Speech Recognition with different architectures and trained with noisy inputs should also keep being explored.

Bibliography

[1] B. Gold, N. Morgan, and D. Ellis, *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*, 2nd ed. New York, NY, USA: Wiley-Interscience, 2011.

[2] C. Ittichaichareon, S. Suksri, and T. Yingthawornsuk, "Speech Recognition using MFCC," *International Conference on Computer Graphics, Simulation and Modeling (ICGSM'2012)*, 2012.

[3] H. Hermansky, "Perceptual Linear Predictive (PLP) Analysis of Speech," *Journal of the Acoustical Society of America*, 1990.

[4] F. Allen, E. Ambikairajah, and J. Epps, "Language Identification using Warping and the Shifted Delta Cepstrum," in *2005 IEEE 7th Workshop on Multimedia Signal Processing*, 2005.

[5] M. Westphal, "The Use of Cepstral Means in Conversational Speech Recognition," in *In Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, 1997.

[6] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, 1989.

[7] G. Zavaliagkos, Y. Zhao, R. Schwartz, and J. Makhoul, "A Hybrid Segmental Neural Net/Hidden Markov Model System for Continuous Speech Recognition," *IEEE Transactions on Speech and Audio Processing*, 1994.

[8] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, and T. Sainath, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," *IEEE Signal Processing Magazine*, 2012.

[9] X. Huang, A. Acero, H. W. Hon, and R. Reddy, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall PTR, 2001.

[10] M. Gales and S. Young, "The Application of Hidden Markov Models in Speech Recognition," *Foundations and Trends in Signal Processing*, 2008.

[11] (2018, Jan) Introduction to deep learning. [Online]. Available: www.introtodeeplearning.com

[12] J. Schmidhuber, “Deep Learning in Neural Networks: An Overview,” *Neural Networks*, 2014.

[13] (2017, August) Recurrent neural networks. [Online]. Available: https://www.youtube.com/watch?v=6niqTuYFZLQ&t=1057s

[14] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, 1997.

[15] A. Graves, G. Wayne, and I. Danihelka, “Neural Turing Machines,” *Google DeepMind*, 2014.

[16] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, “End-To-End Memory Networks,” in *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015.

[17] L. Pfeifenberger, T. Schrank, M. Zöhrer, M. Hagmüller, and F. Pernkopf, “Multi-Channel Speech Processing Architectures for Noise Robust Speech Recognition: 3rd CHiME Challenge Results,” in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, December 2015.

[18] K. Kinoshita, M. Delcroix, S. Gannot, E. A. P. Habets, R. Haeb-Umbach, W. Kellermann, V. Leutnant, R. Maas, T. Nakatani, B. Raj, A. Sehr, and T. Yoshioka, “A Summary of the REVERB Challenge: State-of-the-Art and Remaining Challenges in Reverberant Speech Processing Research,” *EURASIP Journal on Advances in Signal Processing*, 2016.

[19] C. Kim, K. K. Chin, M. Bacchiani, and R. M. Stern, “Robust Speech Recognition Using Temporal Masking and Thresholding Algorithm,” in *INTERSPEECH-2014*, 2014.

[20] C. Kim and R. M. Stern, “Power-Normalized Cepstral Coefficients (PNCC) for Robust Speech Recognition,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2016.

[21] T. Athanaselis, S. Bakamidis, G. Giannopoulos, I. Dologlou, and E. Fotinea, “Robust Speech Recognition in the Presence of Noise Using Medical Data,” in *2008 IEEE International Workshop on Imaging Systems and Techniques*, 2008.

[22] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, “Convolutional Neural Networks for Speech Recognition,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2014.

[23] T. Morito, O. Sugiyama, R. Kojima, and K. Nakadai, “Partially Shared Deep Neural Network in Sound Source Separation and Identification Using a UAV-Embedded Microphone Array,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[24] T. Ohata, K. Nakamura, T. Mizumoto, T. Taiki, and K. Nakadai, “Improvement in Outdoor Sound Source Detection Using a Quadrotor-Embedded Microphone Array,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[25] L. Wang and A. Cavallaro, “Microphone-Array Ego-Noise Reduction Algorithms for Auditory Micro Aerial Vehicles,” *IEEE Sensors Journal*, 2017.

[26] T. Tezuka, T. Yoshida, and K. Nakadai, “Ego-Motion Noise Suppression for Robots based on Semi-Blind Infinite Non-Negative Matrix Factorization,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[27] A. Ito, T. Kanayama, M. Suzuki, and S. Makino, “Internal Noise Suppression for Speech Recognition by Small Robots.” in *INTERSPEECH-2005*, 2005.

[28] Audacity. [Online]. Available: www.audacityteam.org

[29] C. Plapous, C. Marro, and P. Scalart, “Improved Signal-to-Noise Ratio Estimation for Speech Enhancement,” *IEEE Transactions on Audio, Speech, and Language Processing*, 2006.

[30] I. Cohen and B. Berdugo, “Noise Estimation by Minima Controlled Recursive Averaging for Robust Speech Enhancement,” *Signal Processing Letters, IEEE*, 2002.

[31] S. Dixit and D. Nagaria, “LMS Adaptive Filters for Noise Cancellation: A Review,” *International Journal of Electrical and Computer Engineering (IJECE)*, 2017.

[32] Mathworks. [Online]. Available: https://www.mathworks.com/help/dsp/ref/dsp.lmsfilter-system-object.html

[33] Audacityteam. [Online]. Available: https://wiki.audacityteam.org/wiki/How_Audacity_Noise_Reduction_Works

[34] C. Gargour, M. Gabrea, V. Ramachandran, and J. Lina, “A Short Introduction to Wavelets and their Applications,” *IEEE Circuits and Systems Magazine*, vol. 9, 2009.

[35] S. Mihov, R. Ivanov, and A. N. Popov, “Denoising Speech Signals by Wavelet Transform,” *Annual Journal of Electronics*, 2009.

[36] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization.” *3rd International Conference for Learning Representations*, 2014.

[37] M. Rubashkin and M. Mollison. (2017) Tensorflow rnn tutorial. [Online]. Available: www.svds.com/tensorflow-rnn-tutorial

[38] M. Schuster and K. K. Paliwal, “Bidirectional Recurrent Neural Networks,” *IEEE Transactions on Signal Processing*, 1997.

[39] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks,” in *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

[40] Crazyflie. [Online]. Available: https://www.bitcraze.io/crazyflie-2

[41] M. Strauss, P. Mordel, V. Miguet, and A. Deleforge, “Dregon: Dataset and Methods for UAV-Embedded Sound Source Localization,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2018)*, 2018.

[42] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: An ASR Corpus Based on Public Domain Audio Books,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[43] ASR Evaluation. [Online]. Available: https://github.com/belambert/asr-evaluation

[44] Thesis’ Code. [Online]. Available: https://github.com/anacatarinaribeirosa/speechrecoguav

[45] A. Graves, A. Mohamed, and G. Hinton, “Speech Recognition with Deep Recurrent Neural Networks,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013.

List of Commands

1. get me the biscuits.
2. get me the original Pringles.
3. get me the lemon Pringles.
4. get me the cheese.
5. get me the seafood.
6. get me the cornflakes.
7. get me the milk.
8. get me the green tea.
9. get me the beer.
10. get me the lemonade.
11. get me the water.
12. get me the yogurt.
13. get me the seven up.
14. get me the spoons.
15. get me the metal cup.
16. get me the red cup.
17. get me the glass.
18. get me the paper cup.
19. get me the mug.
20. bring me water.
21. bring me the coffee.
22. good job.
23. thank you.
24. yes.
25. no.
26. follow me.
27. stop.
28. come with me.
29. go to the living room.
30. go to the bedroom.
31. go to the dining room.
32. go to the kitchen.
33. go to the bathroom.
34. head for the front door.
35. head for the back door.
36. take me to the couch.
37. take me to the dining table.
38. take me to the kitchen table.
39. take me to the stove.
40. take me to the sofa.
41. take me to the armchair.
42. open the door.
43. close the door.
44. take me to the bed.
45. lift me up.
46. lay me down.
47. move forward.
48. move right.
49. move left.
50. move back.
51. move backwards.
52. half speed.
53. full speed.

B

Additional Information on the Commands

Table B.1: SNR for testing commands.

Command	WER	SNR	Command	WER	SNR
1	25%	$18,95dB$	28	0%	$15,80dB$
2	40%	$2,20dB$	29	0%	$-0,77dB$
3	60%	$15,78dB$	30	0%	$12,03dB$
4	25%	$11,18dB$	31	0%	$13,10dB$
5	25%	$22,03dB$	32	0%	$20,87dB$
6	0%	$21,64dB$	33	25%	$20,00dB$
7	0%	$3,88dB$	34	0%	$5,36dB$
8	40%	$8,72dB$	35	0%	$8,18dB$
9	100%	$23,26dB$	36	0%	$24,11dB$
10	0%	$18,76dB$	37	0%	$19,10dB$
11	0%	$14,22dB$	38	0%	$15,51dB$
12	0%	$7,54dB$	39	20%	$6,43dB$
13	50%	$10,22dB$	40	0%	$7,82dB$
14	0%	$2,63dB$	41	0%	$1,63dB$
15	20%	$14,47dB$	42	0%	$11,51dB$
16	40%	$24,77dB$	43	0%	$22,36dB$
17	0%	$9,20dB$	44	20%	$10,32dB$
18	40%	$4,73dB$	45	33,3%	$-0,33dB$
19	0%	$23,33dB$	46	0%	$18,70dB$
20	0%	$1,48dB$	47	0%	$-0,94dB$
21	100%	$18,57dB$	48	0%	$16,67dB$
22	0%	$1,05dB$	49	100%	$1,73dB$
23	0%	$15,27dB$	50	0%	$16,27dB$
24	0%	$-3,98dB$	51	0%	$0,78dB$
25	0%	$-0,41dB$	52	0%	$2,25dB$
26	0%	$-1,13dB$	53	100%	$0,15dB$
27	0%	$1,52dB$			