Speech Recognition for a Small Aerial Robot

Ana Catarina Monteiro Ribeiro de Sá[1],
[1]Department of Electrical and Computer Engineering, Instituto Superior Técnico, Lisbon, Portugal

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.

***Index Terms*—Speech Recognition, Unmanned Aerial Vehicles, Noisy Speech Signal, Least-Mean-Square Adaptive Filter, Ego-noise Cancellation.**

I. Introduction

UNMANNED Aerial Vehicles (UAVs) is a technology that deployed capture of data in areas of difficult access. The merging of Computer Vision with drones gave a big boost to data acquisition and the appearance of new applications but is limited to visual data only. Being able to acquire audio data with an UAV can be another important innovation in these type of systems. One of the reasons why this subject has not been widely studied and developed is due to the challenging problem of filtering not only background noise but most importantly, noise from the drones' engines. This problem becomes more significant when we deal with Micro Aerial Vehicles (MAVs). Due to their dimensions, the high proximity of a microphone to the source of ego-noise is inevitable and can lead to data acquisition where the wanted speech signal is almost completely masked by the noise signal. With this project we want to study the possibility of integrating a Speech Recognition system in a small UAV and propose methods to successfully process and recognize noisy speech signals. Besides enabling the usage of these type of systems in search and rescue situations, the outcomes of this research can also give an important contribution to improve Human Robot Interaction.

With this in mind, the thesis was developed going through a series of goals and tasks present in this article according to the following outline: Section II gives some detailed explanation on basic concepts of Signal Processing, Speech Recognition, Neural Networks and UAVs. In section III we present the two approaches to solve the problem, filtering first the noisy signals that are to be fed to a Speech Recognition system and train a Neural Network for Speech Recognition with noisy input signals. For the first approach we focus in 5 filters: Wiener Filter, Least-Mean-Square Adaptive Filter, Noise Gate, Motion-Dependent Spectral Subtraction and Discrete Wavelet Transform. For the second approach, we explore the usage of a Recurrent Neural Network with Connectionist Temporal Classification to directly convert the noisy speech signal into the transcription of the speech. The results obtained for the previously mentioned implementations are in section IV and an analysis is provided. Finally, in section V we present the conclusions taken from this research together with suggestions for work to be done in the future on this subject.

Corresponding author: A. Sá (email: ana.catarina.sa@tecnico.ulisboa.pt).

II. Overview

A. Signal Enhancement and Filtering for Automatic Speech Recognition

We chose to study two different approaches to solve the problem of this research. We can either filter the noisy speech input and then feed the result to an Automatic Speech Recognition (ASR) system or we can teach the recognizer how to understand noisy inputs.

Noise filtering when using drones is a challenging task not only due to ego-noise of the drone but also to the high variability of background noises such as winds, cars' horns or the sound of televisions. Background noise subtraction has been mainly addressed in studies on sound source localization. Examples of proposed methods for denoising are Adaptive Beamforming [1], Blind Source Separation [2] and statistical Room Impulse Reverberation (RIR) modeling [3]. Noise filtering when focusing on the drone's ego-noise is a challenging task due to the high power of noise compared with the power of the speech signal. Most of the algorithms that focus on noise cancellation perform Spectral Subtraction. For these type of filters there's usually the necessity of working in the frequency domain and this is achieved by applying a discrete Short-Time

Fourier Transform (STFT) to an audio signal as it is written in equation 1.

$$X(k,p) = \sum_{n=-\infty}^{\infty} x[n]w[n-p]e^{-jkn} \quad (1)$$

where $X(k,p)$ is the STFT of the discrete signal $x[n]$ at frequency k and short-time bin p passing through a window $w[n]$.

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. The high variability of characteristics like speaker dependency and surrounding conditions in speech is one of the biggest barriers for the development of robust ASR systems [4].
A traditional ASR system is composed by several modules: Feature Extraction, Pronunciation Model, Acoustic Model, Language Model and Decoder. Feature Extraction is the first step and has the goal of optimally choose acoustic features of the speech signal in order to reduce model complexity while maintaining relevant linguistic information. The Mel-Frequency Cepstral Coefficients (MFCCs) 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 MFCCs, m, and frequency in Hertz,f, is computed through equation 2.

$$m = 2595 \times log_{10}\left(1 + \frac{f}{700}\right) \quad (2)$$

B. Neural Networks for Speech Recognition

Deep Learning (DL) has been having a very important role in the improvement of ASR systems and, although the goal is to apply DL to the whole ASR process, Speech Recognition systems have not achieved that level yet and there is still the necessity of making feature extraction in the state-of-the-art models.
Two types of Neural Networks (NNs) are being explored for Speech Recognition. One is based on the idea of analysing the speech spectrogram as an image and use Convolutional Neural Networks [5], the most used method for image classification, to make speech transcriptions. The other approach is to use Recurrent Neural Networks (RNN) to compute the phonemes of a speech signal from the features extracted in the beginning of the ASR system. The usage of Recurrent NN is supported by the fact that these NNs have flexibility on the quantity of inputs and outputs of the system and exhibit temporal dynamic behavior. 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. Since it is so difficult to store so much information, Long-Short Term Memory (LSTM) networks [6] were introduced to deal with this problem and have shown to perform better than conventional RNN.

C. Unmanned Aerial Vehicles

Nowadays UAVs can have such small measures that they can even fit in a person's hand. The starting point of this project was to use an implementation previously developed of a single-channel microphone in a Crazyflie as the one we can see in Figure 1. Due to the poor quality of the recordings, we opted to use the DREGON dataset [7]. This dataset is composed by motors' noise recordings with an 8-channel microphone array of the MikroKopter quadrotor UAV in Figure 2.

Fig. 1: Crazyflie 2.0.

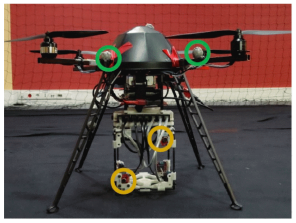

Fig. 2: MikroKopter quadrotor with 8-channel microphone array.

With manipulation of the data through the program Audacity [8] we converted the audio into single-channel and mixed the tracks in order to simulate combinations of different speeds between the 4 motors.

III. Implementation

A. Filtering Algorithms

The first approach followed was to filter the noisy speech signals before feeding them to an ASR system. The 5 filters chosen were Wiener Filter, Least-Mean-Square (LMS) Adaptive Filter, Noise Gate, Motion-Dependent Spectral Subtraction (MDSS) and Discrete Wavelet Transform (DWT).

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 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 the sum of the clean speech signal, $s[n]$, and the noise-only signal, $n[n]$. To obtain an estimation of the clean speech we will follow the methods used in [9] that starts 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 $a\ priori$ Signal-to-Noise Ratio (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 and 4.

$$S\hat{N}R_{prio}(k,p) = \frac{E\left[|S(k,p)|^2\right]}{E\left[|N(k,p)|^2\right]} \tag{3}$$

$$S\hat{N}R_{post}(k,p) = \frac{|X(k,p)|^2}{E\left[|N(k,p)|^2\right]} \tag{4}$$

where $E\left[.\right]$ represents the expectation operator. The spectral gain can be then obtained through equation 5 in which function $g(.)$ is a gain function proposed in methods like power spectral subtraction or wiener filter.

$$G(k,p) = g(S\hat{N}R_{prio}(k,p), S\hat{N}R_{post}(k,p)) \tag{5}$$

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 6

$$\hat{S}(k,p) = G(k,p)X(k,p) \tag{6}$$

Through the reading of [9] we conclude that the computation of the minimum of $E\{(\hat{S}(k,p) - S(k,p))^2\}$ leads to equation 7

$$\begin{aligned} G(k,p) =& \frac{E\left[|S(k,p)|^2\right]}{E\left[|S(k,p)|^2\right] + E\left[|N(k,p)|^2\right]} \\ =& \frac{S\hat{N}R_{prio}(k,p)}{1 + S\hat{N}R_{prio}(k,p)} \end{aligned} \tag{7}$$

Looking at equation 3 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:

$$S\hat{N}R_{post}(k,p) = \frac{|X(k,p)|^2}{\hat{\gamma}_n(k,p)} \tag{8}$$

$$\begin{aligned} &S\hat{N}R^{DD}_{prio}(k,p) = \\ &\beta\frac{|\hat{S}(k,p-1)|^2}{\hat{\gamma}_n(k,p)} + (1-\beta)P\left[S\hat{N}R_{post}(k,p) - 1\right] \end{aligned} \tag{9}$$

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 through minima controlled recursive averaging approach [10] and $P\left[.\right]$ represents the half-wave rectification.

Once we have obtained G_{DD} through equation 7 we can now advance to the TSRT method to make the prediction of speech. The new $a\ priori$ SNR is given by

$$S\hat{N}R^{TSRT}_{prio}(k,p) = \frac{|G_{DD}(k,p)X(k,p)|^2}{\hat{\gamma}_n(k,p)} \tag{10}$$

Applying the result of equation 10 to equation 7, the estimation of the clean speech can finally be computed through equation 11.

$$\hat{S}(k,p) = G_{TSRT}(k,p)X(k,p) \tag{11}$$

When listening the resulting signal, $\hat{s}[n]$, a distortion is clear. This 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. We can obtain the final estimation of the clean speech with restore of the harmonics applying equations 12 to 15.

$$s_{harmo} = NL(\hat{s}(t)) = max(\hat{s}(t), 0) \tag{12}$$

$$\begin{aligned} &S\hat{N}R^{HRNR}_{prio}(k,p) = \\ &\frac{G_{TSRT}(k,p)|\hat{S}(k,p)|^2 + (1 - G_{TSRT}(k,p))|S_{harmo}(k,p)|^2}{\hat{\gamma}_n(k,p)} \end{aligned} \tag{13}$$

$$G_{HRNR}(k,p) = \frac{S\hat{N}R^{HRNR}_{prio}(k,p)}{1 + S\hat{N}R^{HRNR}_{prio}(k,p)} \tag{14}$$

$$\hat{S}(k,p) = G_{HRNR}(k,p)X(k,p) \tag{15}$$

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 [11]. 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.

Specifically for this project what we want to implement is Adaptive Noise Cancelling. Figure 3 shows the block diagram of the algorithm that will be implemented.

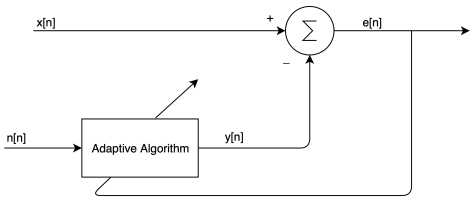

Fig. 3: Block diagram of Adaptive Noise Cancelling.

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 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, 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 [12]. Equations 16 to 20 show the computations required to obtain the prediction of the speech-only signal.

$$y[n] = \boldsymbol{w}^T[n-1]\boldsymbol{u}[n] \tag{16}$$

$$\boldsymbol{u}[n] = [n[n] \quad n[n-1] \quad ... \quad n[n-order]] \tag{17}$$

$$\boldsymbol{w}[n] = \alpha\boldsymbol{w}[n-1] + \mu e[n] sign(\boldsymbol{u}[n]) \tag{18}$$

$$sign(\boldsymbol{u}[n]) = \begin{cases} -1 & \text{if } \boldsymbol{u}[n] < 0 \\ 0 & \text{if } \boldsymbol{u}[n] = 0 \\ 1 & \text{if } \boldsymbol{u}[n] > 0 \end{cases} \tag{19}$$

$$e[n] = x[n] - y[n] \tag{20}$$

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. The 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 $\in \{2, 5, 7, 12, 20\}$.

3) Noise Gate

Noise Gate [13] is an algorithm that uses Fourier analysis. The first step of the algorithm is applying 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 decibels (dBs) 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 21 and 22.

$$N(k,p) = STFT\{n[n]\} \tag{21}$$

$$N_{dB}(k,p) = 10log_{10}(N(k,p)) \tag{22}$$

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 23.

$$\lambda(k) = \overline{N_{dB}}(k) + n \times \sigma_k \tag{23}$$

After having the threshold values for each frequency we can now compute the power spectrum in dBs of the whole noisy signal, $x[n]$, as it was done in 21 and 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 24 is then applied.

$$G_{dB}(w,t) = \begin{cases} 0dB & \text{if } X_{dB}(k,p) \geq \lambda(k) \\ min(X_{dB}(k,p)) & \text{if } X_{dB}(k,p) < \lambda(k) \end{cases} \tag{24}$$

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 25 we obtain the predicted clean speech in dB.

$$\hat{s}_{dB}(k,p) = X_{dB}(k,p) + \tilde{G}_{dB}(k,p) \tag{25}$$

The denoised signal, $\hat{s}[n]$, is recovered by converting $\hat{s}_{dB}(k,p)$ to linear followed by the computation of the Inverse Short-Time Fourier Transform (ISTFT).

4) Motion-Dependent Spectral Subtraction

In [14] we find a filtering approach based in spectral subtraction like the Noise Gate. 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 [14] of filtering magnitude only 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 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 26 to divide the noise components in each set of frequencies.

$$m(l) = \sum_{f=f_{lo}(l)}^{f_{hi}(l)} W_l(f)N(f) \tag{26}$$

The window applied is defined in equation 27.

$$W_l(f) = \begin{cases} \frac{f - f_{lo}(l)}{f_c(l) - f_{lo}(l)} & \text{if } f_{lo}(l) \leq f \leq f_c(l) \\ \frac{f_{hi}(l) - f}{f_{hi}(l) - f_c(l)} & \text{if } f_c(l) \leq f \leq f_{hi}(l) \\ 0 & \text{otherwise} \end{cases} \quad (27)$$

In equations 26 and 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:

$$\overline{N}_l = \frac{m(l)}{\sum_{f=f_{lo}(l)}^{f_{hi}(l)} W_l(f)} \quad (28)$$

Once we have estimated the average noise, 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 29

$$\tilde{S}_l(f) = \begin{cases} X(f) - \bar{N}_l & \text{if} |X(f)|^2 - \alpha|\bar{N}_l|^2 \geq \beta \bar{N}_l^2 \\ 0 & \text{if} |X(f)|^2 - \alpha|\bar{N}_l|^2 < \beta \bar{N}_l^2 \end{cases} \quad (29)$$

where $\alpha \geq 1$ and $0 < \beta < 1$. To recover the speech-only spectrum we have to apply again the window computed in 27 to the predicted speech obtained in 29.

$$S(f) = \frac{\sum_{l=1}^{l\_max} W_l(f) \tilde{S}_l(f)}{\sum_{l=1}^{l\_max} W_l(f)} \quad (30)$$

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 DFT.

5) Discrete Wavelet Transform

The last filter implemented is based on the Discrete Wavelet Transform (DWT) [15]. In order to use the DWT for audio denoising we have to follow the steps in equations 31 to 35.

$$x(t) = s(t) \otimes n(t) \quad (31)$$

$$y = W(x) \quad (32)$$

$$z = D(y, \lambda) \quad (33)$$

$$\hat{s} = W^{-1}(z) \quad (34)$$

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 4 shows the process of a 3 level decomposition.

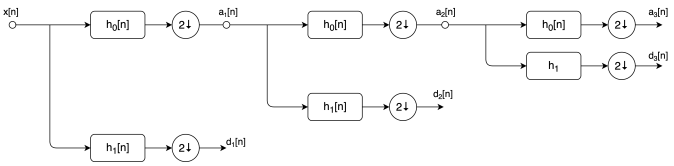

Fig. 4: Level 3 Wavelet Transform Diagram.

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 [16]. The DWT is defined in equation 35:

$$DWT = \int_{-\infty}^{+\infty} x(t)\psi_{j,k}(t) \quad (35)$$

where

$$\psi_{j,k}(t) = \frac{1}{\sqrt{s^j}} \psi\left(\frac{t - k2^j}{2^j}\right) \quad (36)$$

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 [16]. 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. To define the threshold we opted to use Universal Threshold specified in equation 37. After that, the soft threshold described in 38 is applied.

$$\lambda = \sqrt{2ln(N)} \quad (37)$$

$$z = \begin{cases} sign(x)(x - \lambda) & \text{if } x > \lambda \\ 0 & \text{if } x \leq \lambda \end{cases} \quad (38)$$

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

Finally the last step is to reconstruct the signal performing the Inverse DFT through the original approximation coefficient and the estimated detail coefficients by performing the reverse process specified in Figure 4.

6) ASR System

After applying the filter to the noisy speech, we used Google Cloud's Speech-to-Text API to obtain the speech transcriptions.

B. Feedforward Neural Network

It is easy to understand that the motor's velocity is going to influence the noise profile. The conjugation of different motors speeds in the 4 motors will lead to numerous noise profiles. In [14] the idea of relating the motors' speeds to the 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 Feedforward NN to the filter previously studied that obtains the best results. To find the relation between the velocities and the filter's coefficients start by defining the training process. The inputs, x, are first passed through a linear layer and then by a non-linear activation function described in 39 and 40, respectively.

$$\hat{y} = xA^T + b \tag{39}$$

$$LeakyRELU(\hat{y}) = \begin{cases} x & \text{if } x \geq 0 \\ 0,01x & \text{if } x < 0 \end{cases} \tag{40}$$

where A and b are the parameters computed 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 41:

$$l_n = (y_n - \hat{y}_n)^2 \tag{41}$$

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 Adam [17] optimization model so we can retrieve the appropriate weights.

Repeating the described process several times leads to an optimization of the prediction made by the Neural Network. In the next section 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.

C. Neural Network for Speech Recognition

The last method we want to explore is to use only a Neural Network implemented for Speech Recognition that is trained with noisy inputs.
In order to quickly test this approach, we are using an already existent Neural Network [18]. 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 [19] 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 [20].

IV. Results

In this section we show the results obtained for each of the approaches explained in the previous section. We settled a list of 53 commands with 198 words in total for testing. Word Error Rate (WER) is the chosen metric to evaluate the performance of the systems. WER is computed according to equation 42

$$WER = \frac{S + D + I}{N} \tag{42}$$

where S is the number of substitutions, D the number of deletions and I the number of insertions in the hypothesis and N the total number of words in the reference. In order to test the robustness of the systems against different levels of noise, we changed the power of the clean speech signals before mixing with noise-only signals so we would obtain the levels of SNR of $-10dB, -5dB, 0dB, 5dB, 10dB$.

A. Speech Recognition without Filtering

1) DREGON dataset

To analyze the decrease of performance of the Speech Recognizer we started by recording the commands with a 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 test performed to the DREGON dataset without filtering is when we added the noise-only files of this dataset to the speech only audio files. Table I shows the WER values obtained for the different SNRs values that we are testing.

TABLE I: WER for no filtering in DREGON dataset.

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

As it is clear now, the values of WER decrease when the SNR is increased. 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.

B. Speech Recognition with Filtering

The results of the previous section support the purpose of this research. 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 is the only dataset used.

The WER values for the 5 filters can be seen in Table II.

TABLE II: WER for all the filters and for different SNR values.

$SNR(dB)$	WF	LMS	NG	MDSS	DWT
-10	99,492%	4,040%	96,954%	93,939%	99,495%
-5	96,954%	3,535%	89,848%	77,273%	94,949%
0	89,848%	3,030%	76,142%	52,525%	82,323%
5	74,619%	5,584%	59,391%	43,939%	60,606%
10	59,898%	4,061%	43,655%	30,808%	49,495%

1) Wiener Filter

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. 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.

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 led us to settle with the parameters $\{order = 5, \mu = 0.05\}$.
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 that decrease 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 paper.

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 23, we applied the values 1, 1,5 and 2 to n. The best WER values obtained were for $\sigma = 1$ therefore we present the values in table II.
Once again we detect the relation mentioned before between the WERs and the SNR values. The resulting audio show the worst performance in 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 III since we only decrease the noise and we maintain the speech. There is almost no distortion in the listened commands but, as it is clear from the results, there is still a lot of interference by the motors' noise.

4) Motion-Dependent Spectral Subtraction

For the MDSS filter we set the values of α and β to 2 and $0,5$, respectively.
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 original noisy 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.

5) Discrete Wavelet Transform

For this last filter we chose 17 levels of decomposition.
Looking at the numbers it is clear that the algorithm doesn't perform good, specially for the cases 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. We 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.

6) Conclusions

Based on the results obtained in the previous subsections, we elaborated the graphic in figure 5 showing the performance of the systems along the SNR values.
Three of the filters were not able to improve the performance of the ASR and even resulted in a higher WER compared to no filtering. 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.
The results for the MDSS algorithm are extremely close to the no filtering results. This shows that the statement done in subsection 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.
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.

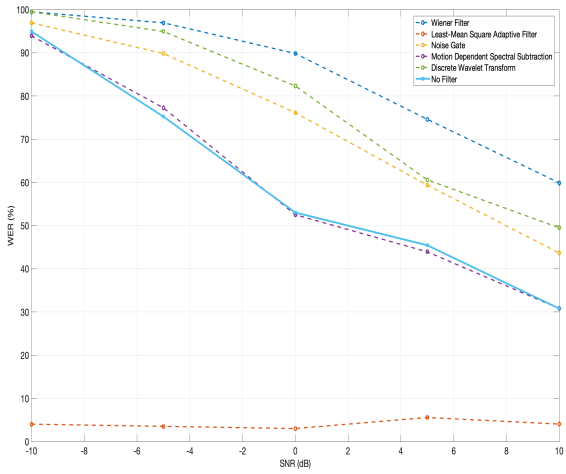

Fig. 5: Comparison of results between original signal and all filtered signals for all the SNR values.

C. Speech Recognition with Filtering using Neural Networks

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 output is 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.

$$cost = \sum_{n=1}^{N} l_n \tag{43}$$

In equation 43, l_n is as defined in equation 41 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, $N = 1596$. By observing the cost associated to the number of epochs when we have different hidden layers' dimensions, the only information we could retrieve is that in less than 500 epochs the NN converges to the best predictions. When we increase the number of hidden layers, the number of epochs necessary to converge for the best result is smaller. Also, we analyzed the cost after stabilization associated to each NN when we change the number of hidden layers and the layers' dimensions. The values obtained from the worst case to the best case had a difference of $0,0287$. We saw a slight improve of the cost when we increase the number of hidden layers. We opted to implement the NN with 2 hidden layers with size 10. With these parameters settled we will present in table III the performance of the ASR system when we directly apply the coefficients computed through the NN to the filtering process.

TABLE III: WER for filtering with LMS Adaptive Filter using a Neural Network.

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

The results obtained are still very close to the optimal outcome (0%). We can conclude that it is possible to use a NN to relate the motors' speeds information to the needed filter coefficients.

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 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. A difference of $0,5dB$ between the two signals resulted in good WER values. An increase of $0,5dB$ is actually translated into a signal with power $1,12$ times higher than the power of the one with lower 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, increasing the distance of one of the microphones to the noise source not only we decrease the power of noise but most likely we will be increasing the power of speech due to approximation of the microphone to the speech source. We will now proceed to the testing using the difference of SNR of $0, 5dB$. The results can be seen in table IV.

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

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

Figure 6 shows the comparison between all the implementations with the LMS Adaptive Filter.

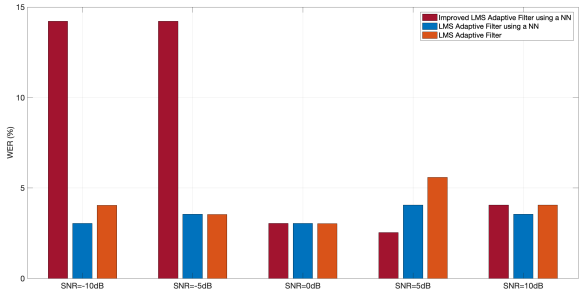

Fig. 6: Comparison between results from all the systems using LMS Adaptive Filtering.

Figure 6 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.

3) Conclusions

In figure 7 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.

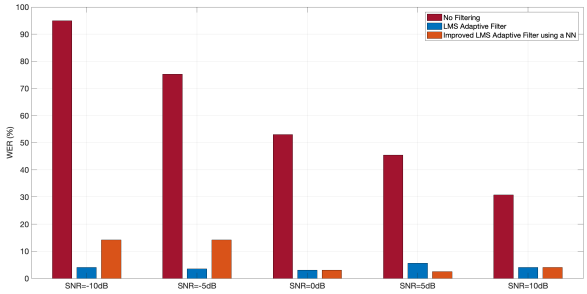

Fig. 7: WER values for best systems in each subsection.

D. Speech Recognition using Recursive Neural Networks

Based on the work developed in [21] 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.

The first step of this implementation is to obtain the MFCCs. 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.

V. Conclusion

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. 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 very close to 0%. A Feedforward Neural Network developed to predict the filter's coefficients based on the information of the speeds of each motor also showed to work. The fact that the denoise is made in real time and not adaptively enabled the possibility of matching or even improving the previously obtained WER. Finally, in order to adapt the system to a real-life situation, 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 previously proposed system.
Using LMS Adaptive Filtering and a Neural Network for filter's coefficients prediction is robust against different values of SNR of the inputs which is the main challenge to overcome in the problem being studied.

A. Future Work

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 the theoretical value presented of $\Delta SNR = 0,5dB$ between the two input signals is enough and achievable.
Since NNs 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.

References

[1] L. Pfeifenberger, T. Schrank, M. Zohrer, M. Hagmüller and F. Pernkopf, *Multi-channel speech processing architectures for noise robust speech recognition: 3rd CHiME challenge results*. 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), 2015.

[2] T. Tezuka, T. Yoshida and K. Nakadai, *Ego-motion noise suppression for robots based on Semi-Blind Infinite Non-negative Matrix Factorization*. 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014.

[3] K. Kinoshita, M. Delcroix, S. Gannot, E.A.P. Habets,E. A. and 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.

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

[5] 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.

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

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

[8] Audacity, www.audacityteam.org.

[9] 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.

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

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

[12] Mathworks, www.mathworks.com/help/dsp/ref/dsp.lmsfilter-system-object.html, 2012.

[13] Audacityteam, wiki.audacityteam.org/wiki/How_Audacity_Noise_Reduction_Works.

[14] A. Ito, T. Kanayama, M. Suzuki and S. Makino, *Internal noise suppression for speech recognition by small robots*. INTERSPEECH-2005, 2005.

[15] C. Gargour, M. Gabrea, V. Ramachandran and J. Lina, *A short introduction to wavelets and their applications*. IEEE Circuits and Systems Magazine, 2009.

[16] S. Mihov, R. Ivanov and A. N. Popov, *Denoising Speech Signals by Wavelet Transform*. Annual Journal Of Electronics, 2009.

[17] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization.*. 2014.

[18] M. Rubashkin and M. Mollison, *TensorFlow RNN Tutorial*. www.svds.com/tensorflow-rnn-tutorial, 2017.

[19] M. Schuster and K. K. Paliwal, *Bidirectional recurrent neural networks*. IEEE Transactions on Signal Processing, 1997.

[20] A. Graves, S. Fernández, F. Gomez and J. Schmidhuber, *Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks*. New York, USA: ACM, 2006.

[21] A. Graves, A. Mohamed and G. Hinton, *Speech recognition with deep recurrent neural networks*. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013.