אוניברסיטת בר־אילן
Bar-Ilan University

הפקולטה להנדסה

המעבדה לעיבוד אותות

מחולל פרויקטים במעבדה לעיבוד אותות

מאור יבדייב בסקיס

עומר פנחס

פרויקט שנה ד' לקראת תואר ראשון בהנדסה

מנחה: ד"ר עופר שוורץ

מנחה אקדמי: פרופ' שרון גנות

אוקטובר 2021

Wind Noise Reduction From Speech Using Spectral Techniques

Maor Yevdayev Baskis, Omer Pinhas
Dr. Ofer Schwartz, Prof. Sharon Gannot

Bar-Ilan University, Israel

Abstract

The noise induced by the wind can be a severe problem in our daily lives, e.g., for telephone calls, for video recording, etc. For such applications, it is rather complicated to add mechanical wind reduction devices. The existence of advanced processors in modern mobile phones allows us to reduce this noise using digital signal processing techniques. This project suggests an algorithm to reduce wind noise from speech, in real time, using a single microphone. For achieving that, we are required to estimate the wind noise power spectral density (PSD). This task is not simple - as common noise reduction methods might fail due to the non-stationary characteristics of wind, special methods are required for this type of problem. For real-time wind noise reduction, the system will segment the audio signal and apply the algorithm on each segment, in the frequency plain.

The first step towards reduction of an undesired wind noise in a speech signal, is detection of segments with wind noise. In this project, 3 different wind noise detection algorithms are proposed, and are combined to a single improved noise detector. All of these methods are exploiting the low-pass-like characteristics of the wind. The second step is the decision rule - it must be decided whether it should apply wind reduction to the signal or not. There are 3 possibilities - wind noise alone, a mixture of wind noise and speech, or no wind noise at all. The third step, is the estimation of the wind noise PSD, which is achieved by exploiting the harmonic structure of the human voice. The fourth step is filtering the signal applying the casual Wiener filter. Finally, the signal is reconstructed using weighted overlap-add (WOLA).

Contents

1 Introduction **2**
- 1.1 Wind Noise Characteristics . . . 2
- 1.2 Different Wind Reduction Methods . . . 3
  - 1.2.1 Morphological Approach . . . 3
  - 1.2.2 Short Term Mean . . . 4
  - 1.2.3 Template Spectrum Combination . . . 4
  - 1.2.4 Attentive Neural Wind Suppression . . . 4

2 System Overview **5**

3 Wind Noise Detection **6**
- 3.1 Spectral Centroid (SC) . . . 6
- 3.2 Zero Crossing Rate (ZCR) . . . 6
- 3.3 Negative Slope Fit (NSF) . . . 7

4 Decision Rule **9**

5 Wind Noise PSD Estimation **10**
- 5.1 Pitch Frequency Estimation . . . 10
- 5.2 Binary Mask . . . 10
- 5.3 Estimation . . . 11

6 Evaluation **12**
- 6.1 Experimental Setup . . . 12
- 6.2 Results . . . 13
- 6.3 Conclusions . . . 15

Appendices **18**

A Code **19**
- A.1 Main . . . 19
- A.2 Functions . . . 22
  - A.2.1 STFT . . . 22
  - A.2.2 ISTFT . . . 23
  - A.2.3 ZCR . . . 23
  - A.2.4 NSF . . . 24
  - A.2.5 SSC . . . 24
  - A.2.6 Decision . . . 24
  - A.2.7 Pitch est . . . 24
  - A.2.8 Noise interpolation . . . 24
  - A.2.9 calc . . . 25
  - A.2.10 Wiener . . . 25

Chapter 1

Introduction

Nowadays, mobile communication is used in almost every situation. However, the benefit of more mobility results in more difficult acoustic conditions. The quality and intelligibility of the recorded speech signal may be affected by additive noise signals. The application of physical wind shields to avoid noise generation acoustically is not practical when the design of mobile equipment, such as mobile phones or hearing aids, gets smaller and thinner by the day. As a result, real-time signal processing is required to remove wind noise from a speech signal, for example, in a phone call. The most crucial steps in doing so - are detection of wind, and PSD estimation. On this project we will show our proposal to reduce wind noise from speech. Our proposed wind detection methods exploit the low-pass characteristics of the wind, explained in chapter 3, while our PSD estimator exploits the harmonic structure of the human speech, explained in chapter 5.

There are various approaches for the wind noise reduction problem. One approach, which we have chosen - is detection of wind noise, estimating the PSD of the wind, and filtering. This approach requires us to find different methods for wind noise detection, and a method to estimate the PSD. Our proposed solution contains 3 wind detection methods as presented in chapter 3, and other 2 methods we have not used are presented in Sec. 1.2. The proposed algorithm is checked using the PESQ and STOI speech-intelligibility measures, for various signal to noise combinations and various types of wind, using the "clean" speech as a reference. The data used for this project is a 20 minute long recording of different people speaking, one at a time, recorded with a single microphone and a 20 minute recording of wind using a single microphone.

1.1 Wind Noise Characteristics

In terms of temporal and spectral characteristics, wind noise varies significantly from other common noise signals. Turbulences in an air stream close to the microphone produce the acoustic signal of wind noise, which results in a non-stationary low frequency noise signal. The spectral energy distribution is a unique feature of wind noise. The spectrum is constant at low frequencies (< 30 Hz), and decays somehow exponentially at a higher frequency band. The spectrum of speech signals, however, differs greatly from this low pass behavior. The histogram shown in Fig. 1.1 illustrates the above.

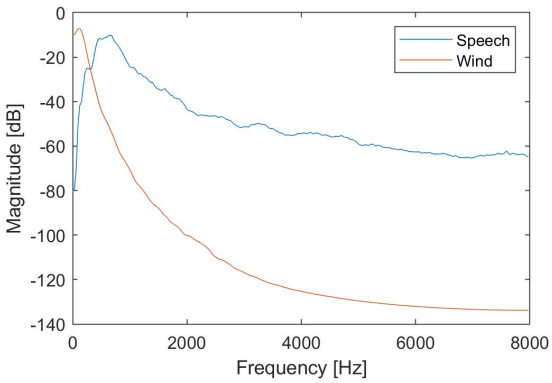

Figure 1.1: Histograms of Speech and Wind

For this reason, we will divide the signal into 3 areas - **(I)** Wind noise alone. **(II)** A mixture of wind noise and speech. **(III)** No wind noise. Therefore, the decision will be either **(I)**, **(II)** or **(III)**.

1.2 Different Wind Reduction Methods

As mentioned before, there are other wind reduction methods we would like to present.

1.2.1 Morphological Approach

The following method, as proposed in [1], uses the morphological characteristics of wind noise and speech in the time-frequency plain. Fig. 1.2 shows a spectrogram of a mixture of speech and wind noise. The human voice consists of pitch frequencies and its harmonies (a fact we will use later) and appears as a set of parallel horizontal lines. Wind noise, however, appears as temporally short vertical lines at low frequencies (as explained in Sec. 1.1). Now, it is proposed to apply the Sobel operator in time direction for an approximate derivative. As shown in Fig. 1.3, the vertical structure of the wind noise is distinguishable, since the derivative of the vocal harmonies is low (the temporal change is small).
Given the temporal derivative, we can now detect wind onsets with a simple algorithm (detecting diagonal lines in the time-frequency domain).

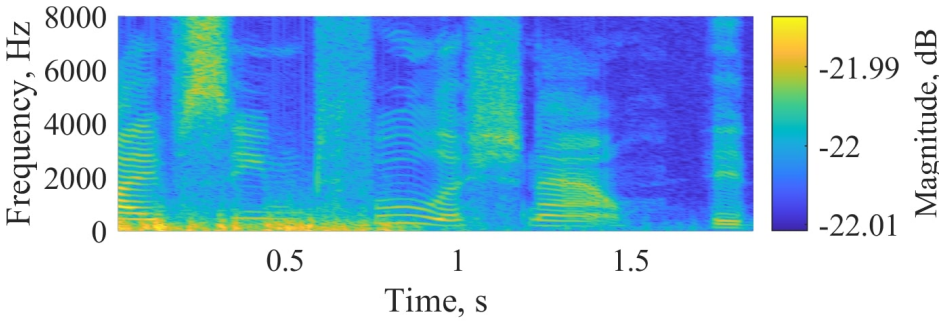

Figure 1.2: A spectrogram of a mixed signal $X(t, f)$

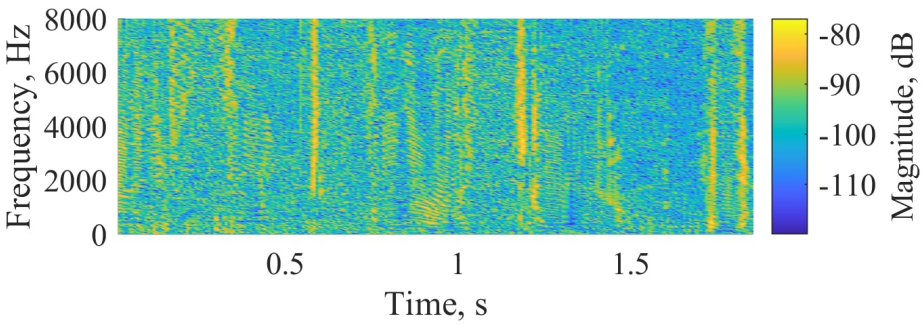

Figure 1.3: $\frac{d}{dt}X(t,f)$

1.2.2 Short Term Mean

Short Term Mean (STM) is a wind detection indicator. The STM, as proposed in [3], is also using the low-pass spectral characteristics of the wind noise. In order to detect the low-frequency parts of the signal, it is suggested to use the short-term mean indicator.

$$I_{STM} = \left| \frac{\sum_{k=1}^{L-1} x_\lambda(k)}{\sum_{k=1}^{L-1} |x_\lambda(k)|} \right| \tag{1.1}$$

The normalization of the term leads to values close to 0 for high frequency components, and values close to 1 for low frequency components - 1 for DC values.

1.2.3 Template Spectrum Combination

Another wind detection indicator - Template Spectrum Combination (TSC), proposed in [3], is using pre-trained notebooks containing speech and noise templates. The noisy magnitude spectrum can be written as:

$$\hat{X}(\lambda,\mu) = \sigma(\lambda)|\tilde{S}_i(\mu)| + (1-\sigma(\lambda))|\tilde{N}_j(\mu)| \tag{1.2}$$

Where $|\tilde{S}_i(\mu)|$ is the speech template and $|\tilde{N}_j(\mu)|$ is the noise template. The noise template is approximated to be $|\tilde{N}_j(\mu)| \propto \frac{1}{f^2}$. The long-term average speech spectrum (LTASS) is used for the speech template. Minimizing the mean square error $||X(\lambda) - \hat{X}(\lambda)||^2$ yields an optimal σ_{opt}. The TSC indicator is denoted by:

$$I_{TSC} = 1 - \sigma_{opt}$$

1.2.4 Attentive Neural Wind Suppression

The following approach, proposed in [7], is meant to reconstruct speech signals damaged by the wind. A relatively low dimensional, shallow neural network is trained to reconstruct the wearer speech signal from wind-corrupted audio specifically in the spectral regions that are most adversely affected by wind noise.

Chapter 2

System Overview

It is assumed $x(k)$ is our noisy signal, which consists of pure speech $s(k)$ and wind noise $v(k)$, i.e. $x(k) = s(k) + v(k)$. Our goal is to estimate $\hat{s}(k)$. The wind noise reduction system is realized as a short-time frequency domain overlap and add structure. The signal is first segmented using a Hamming window with 75% overlap. Using the Fast Fourier Transform (FFT), the frames are transformed into the frequency domain, yielding $X(\lambda, \mu)$ where λ is the frame index and μ is the frequency index. The wind noise PSD $\hat{\Phi}_{vv}(\lambda, \mu)$ is estimated, and a Wiener filter $G(\lambda, \mu)$ is applied for noise reduction. The Wiener filter is given by:

$$G(\lambda, \mu) = \frac{\hat{\Phi}_{ss}(\lambda, \mu)}{\hat{\Phi}_{vv}(\lambda, \mu) + \hat{\Phi}_{ss}(\lambda, \mu)} \tag{2.1}$$

Where $\hat{\Phi}_{ss}(\lambda, \mu)$ is a decision-directed smoothed speech PSD estimation. $\hat{\Phi}_{ss}(\lambda, \mu)$ is calculated by:

$$\hat{\Phi}_{ss}(\lambda, \mu) = \alpha_w |\hat{S}(\lambda, \mu - 1)|^2 + (1 - \alpha_w) max\{\hat{\Phi}_{ss}(\lambda, \mu) - \hat{\Phi}_{vv}(\lambda, \mu), 0\} \tag{2.2}$$

The estimated speech is:

$$\hat{S}(\lambda, \mu) = max\{G(\lambda, \mu), G_{min}\} X(\lambda, \mu) \tag{2.3}$$

G_{min} is the low threshold for our filter - We do not want to hear complete silence at some parts of the estimated signal. The estimated time domain signal $\hat{s}(k)$ is obtained by using the IFFT and weighted overlap-add.

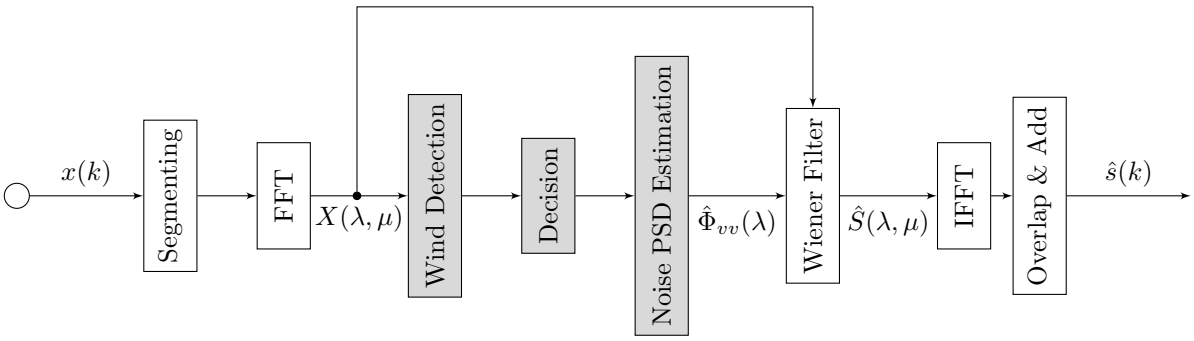

Figure 2.1: The System

Chapter 3

Wind Noise Detection

In order to reduce the wind noise properly without damaging the speech, we need to make a three-way decision rule as explained in Sec. 1.1. For that, we will apply 3 indicators, and combine them to a single decision.

3.1 Spectral Centroid (SC)

For the detection of wind, we will use a 'Spectral Centroid'. The SC, proposed in [5], calculates the spectral "center-of-gravity", or spectral mean value. The SC is defined by:

$$SC(\lambda) = \frac{f_s}{L} \frac{\sum_{\mu=\mu_L}^{\mu_H - 1} \mu \cdot \tilde{\Phi}_{ss}(\lambda, \mu)}{\sum_{\mu=\mu_L}^{\mu_H - 1} \tilde{\Phi}_{ss}(\lambda, \mu)} \tag{3.1}$$

With L being the number of frame indices, M being the number of frequency indices and f_s being the sampling frequency. μ_L is the lowest frequency bin we use here, since the energy of the wind in <40 Hz regions is high - which causes a low frequency SC. By ignoring these DC-like frequencies, we get a real indication regarding the state of the audio. Likewise, μ_H is the highest frequency bin we use. In Fig. 3.1, the histograms of SC of wind noise and speech are shown. About 20 minutes of audio were used for each histogram.

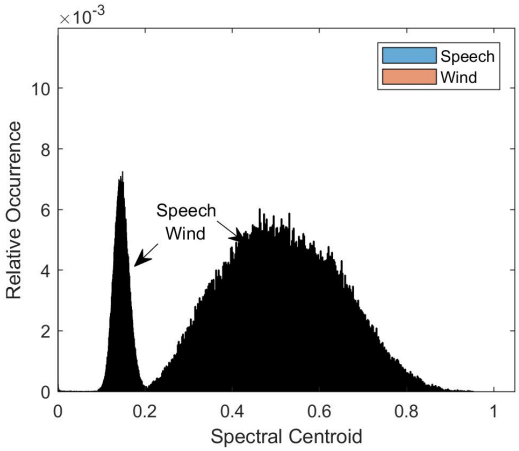

Figure 3.1: Histogram of SC Indicator

3.2 Zero Crossing Rate (ZCR)

The zero crossing rate (ZCR), as proposed in [3], is defined as the number of sign changes in a given signal over a fixed period of time, i.e. the rate at which the signal changes from positive to negative amplitudes or vice versa.

$$ZCR(\lambda) = \frac{1}{L-1}\sum_{l=2}^{L-1} I\{(x(k)\cdot x(k-1) < 0\} \tag{3.2}$$

$I(A)$ equals 1 when A is true, 0 when A is false. Higher frequency signals result in fast changes, therefore many sign changes and a ZCR value close to 1. Lower frequency signals result in a slower changes and a ZCR value close to 0. Dominant frequencies, i.e. high energy spectral components, will mainly affect the ZCR. The ZCR will show us the active frequencies, thus allowing us to determine which of the 3 regions the signal belongs to, as explained in Sec. 1.1. In Fig. 3.2, the histograms of ZCR of wind noise and speech are shown. About 20 minutes of audio were used for each histogram.

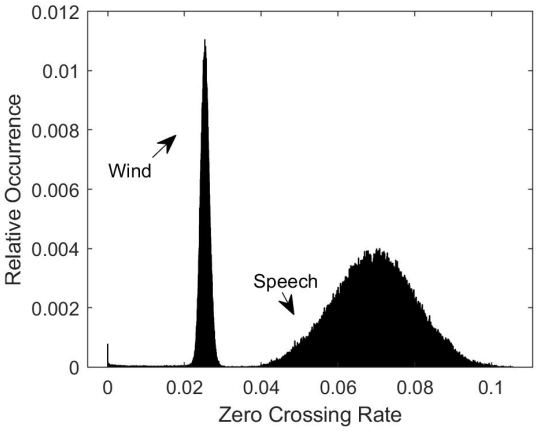

Figure 3.2: Histogram of ZCR Indicator

3.3 Negative Slope Fit (NSF)

The following detector, proposed in [3], is based on the idea that the magnitude of the spectrum of wind noise for certain frequencies, $f < f_{max}$, can be roughly approximated by a linear decay over the frequency:

$$\hat{X}(\lambda) = a_0 \cdot \mu + a_1 \tag{3.3}$$

with μ defined as:

$$\mu = [0, 1, ..., N/2]^T \tag{3.4}$$

The parameter a_0 controls the slope, and a_1 controls the DC and will be denoted as $a = [a_0, a_1]^T$. We will define a matrix, M, to be:

$$M = [1, \mu] \tag{3.5}$$

with 1 being a vector of ones to form a $(N/2+1)\times 2$ matrix. Now, eq. 3.3 can be written as:

$$\hat{X}(\lambda) = M \cdot a \tag{3.6}$$

Because a negative slope is expected, this detector is called "Negative Slope Fit". Minimizing the mean square error yields the Least-Squares estimator:

$$a_{opt}(\lambda) = (M^T M)^{-1} \cdot M^T X(\lambda) \tag{3.7}$$

Now, the NSF indicator is denoted by:

$$NSF(\lambda) = \begin{cases} 1 - \frac{||\hat{X}(\lambda) - X(\lambda)||^2}{||X(\lambda)||^2} & \text{if } a_{1,opt} < 0 \\ 0 & \text{else} \end{cases} \tag{3.8}$$

In Fig. 3.3, the histograms of NSF of wind noise and speech are shown. About 20 minutes of audio were used for each histogram.

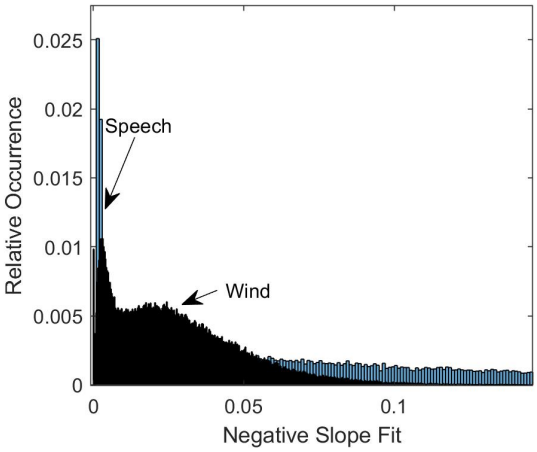

Figure 3.3: Histogram of NSF Indicator

Chapter 4

Decision Rule

In order to decide - **(I)**, **(II)** or **(III)**, we will use the Nayman-Pearson lemma, considering two hypotheses - H_1 = Speech, H_0 = Wind. The vector of the indicators calculated for a segment of data is denoted by $\bar{x}$ (a 3×1 vector, for 3 indicators - SC, ZCR, NSF). The logarithmic likelihood ratio (LLR) will be:

$$LLR(\bar{x}) = log\frac{f(\bar{x}|H_1)}{f(\bar{x}|H_0)}$$

Assuming the indicators have a Gaussian distribution, we get

$$LLR(\bar{x}) = -\frac{1}{2}((\bar{x}-\bar{\mu}_1)^T C_1^{-1}(\bar{x}-\bar{\mu}_1) - (\bar{x}-\bar{\mu}_0)^T C_0^{-1}(\bar{x}-\bar{\mu}_0)) + \frac{1}{2}(log(|C_1|) - log(|C_0|)) \quad (4.1)$$

Where $\bar{\mu}_1, C_1$ are the mean vector and the covariance matrix for speech, and $\bar{\mu}_0, C_0$ are the mean vector and the covariance matrix for wind. These are obtained by applying our 3 indicators on a large signal of pure speech, and a large signal of pure wind.
Since our decision is a 3 way decision, we define γ_1 and γ_2, so that:

$$decision(\bar{x}) = \begin{cases} \textbf{(I)} & \text{if } LLR(\bar{x}) \leqslant \gamma_1 \\ \textbf{(II)} & \text{if } \gamma_1 < LLR(\bar{x}) < \gamma_2 \\ \textbf{(III)} & \text{if } LLR(\bar{x}) \geqslant \gamma_2 \end{cases} \quad (4.2)$$

Where γ_1 and γ_2 are obtained by analyzing big data.

Chapter 5

Wind Noise PSD Estimation

For the estimation of the power spectral density function (PSD) of the wind $\hat{\Phi}_{vv}(\lambda)$, we handle each of the 3 decision regions separately (as defined in Sec. 1.1).
For decision **(I)**, i.e. there is wind and no speech, $\hat{\Phi}_{vv}(\lambda) = X(\lambda)$. For decision **(III)**, i.e. there is no wind, $\hat{\Phi}_{vv}(\lambda) = 0$. For decision **(II)**, i.e. a mixture of speech and wind, we will use a more complex approach for the estimator.
The human voice consists of a pitch frequency and harmonics, which would look like a set of parallel horizontal lines in the time-frequency plane. We will exploit this characteristic of speech in order to remove it and estimate the PSD of the wind.

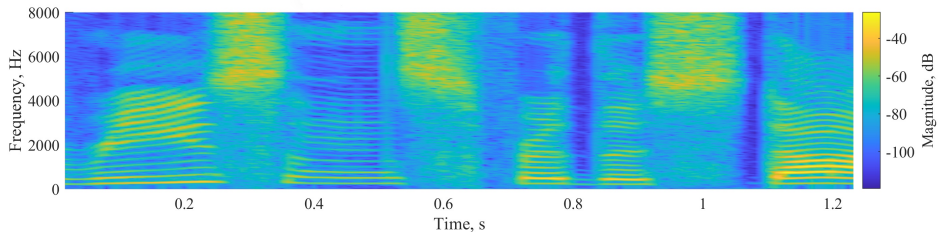

Figure 5.1: Harmonic Structure of Human Speech

5.1 Pitch Frequency Estimation

For the estimation of the pitch frequency of the speech, we will apply a set of masks and find which one fits the best. Assuming there are P different possible pitch frequencies, we will construct a $P \times L$ matrix $F(p,t)$, where L is the number of spatial bins (ranging from f_l to f_h), and a $P \times 1$ vector for the possible pitch frequencies. Each row in F will be constructed as

$$R_p = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & 1 & \dots \end{pmatrix} \tag{5.1}$$

where the amount of zeros between ones is the correspondent pitch frequency in the p-row of the pitch vector.
Applying a Fourier transform yields a new matrix $\tilde{F}(p,f)$. Now, for each frame the pitch frequency estimator is denoted by

$$\mu_0 = arg\max_p \sum_{f_l<f<f_h} \tilde{F}(p,f)X(f) \tag{5.2}$$

5.2 Binary Mask

For the estimation of the wind noise PSD, as suggested in [4], we will use a binary mask.

$$G_{BM}(\mu) = \begin{cases} 1 & \text{if } \mu \in M \\ 0 & \text{else} \end{cases} \tag{5.3}$$

With

$$M = \{\mu | \forall n \in \mathbb{N} : n\mu_0 - \Delta\mu \leqslant \mu \leqslant n\mu_0 + \Delta\mu\} \tag{5.4}$$

Now, $X(\lambda)$ is multiplied by the binary mask, which yields a PSD with "holes" in it. These "holes" replace the vocal harmonies, and by doing this, we "eliminate" the speech.

5.3 Estimation

After we have "eliminated" the speech from our PSD, we need to "fill up the holes". We will use linear interpolation between 2 points - the edges of the holes, and this will be done for each and every "hole" we created.

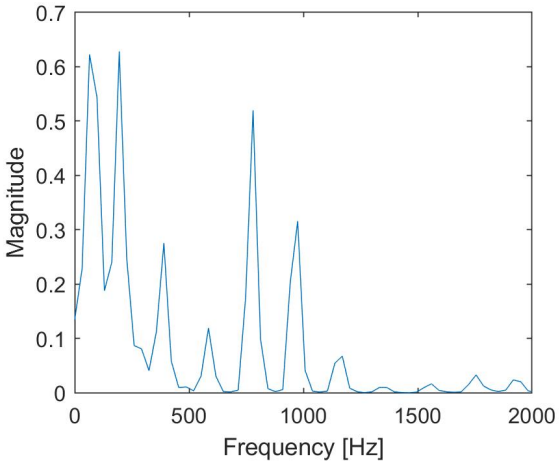

Figure 5.2: PSD of a mixed signal

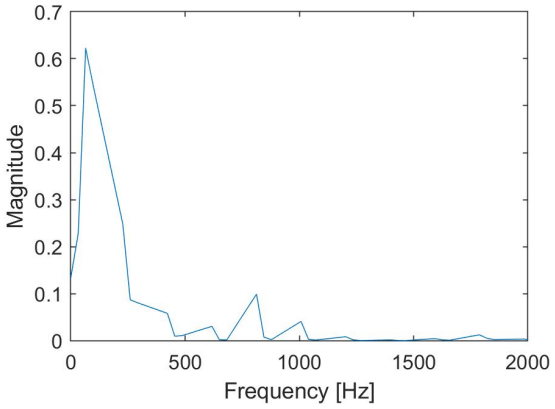

Figure 5.3: Estimated wind noise PSD

Chapter 6

Evaluation

6.1 Experimental Setup

Chapter 2	Sampling Frequency f_s	$16k$Hz
	Segmenting Window	512 Hamming
	Overlap	75%
	FFT Length	512
	α_w	0.95
	G_{min}	between 0.01 and 0.1
Chapter 3	μ_L	100 Hz
	μ_H	2000 Hz
Chapter 4	γ_1	80 Hz
	γ_2	450 Hz
	f_{max}	1000 Hz
Chapter 5	Possible Pitch Freq.	18, between 80 and 250 Hz
	$\Delta\mu$	60 Hz
	f_l	330 Hz
	f_h	6600 Hz

Table 6.1: Simulation Parameters

The performance of the proposed system was evaluated with wind noise recordings and speech recordings, 20 minutes for each. The recordings of the wind noise were made using a mobile phone, in a room with 3 rotating fans. The measures of the room are $3.5m \times 2.6m \times 2.1m$. The single-channel speech recordings were taken from LibriSpeech ASR Corpus ([6]), with one speaker at a time. For the simulation with different wind types, the wind used is a simulated wind using a MATLAB code. The parameters were not changed between simulations. The code was written in MATLAB, and every simulation was also done in MATLAB.

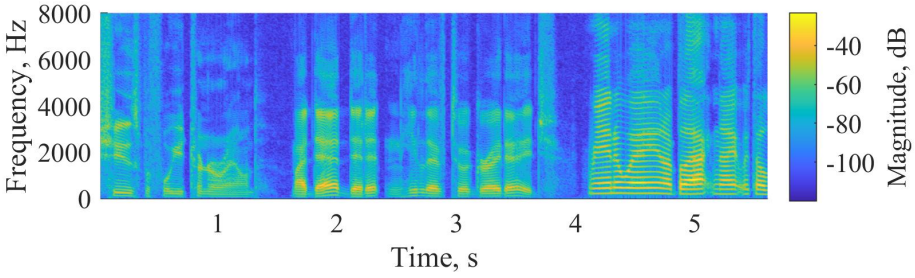

Figure 6.1: Sample of the speech signal used for continuous wind simulation

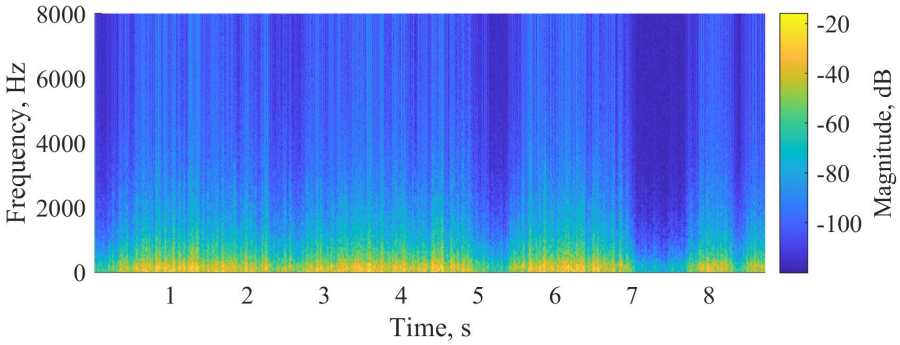

Figure 6.2: Sample of the wind signal used for continuous wind simulation

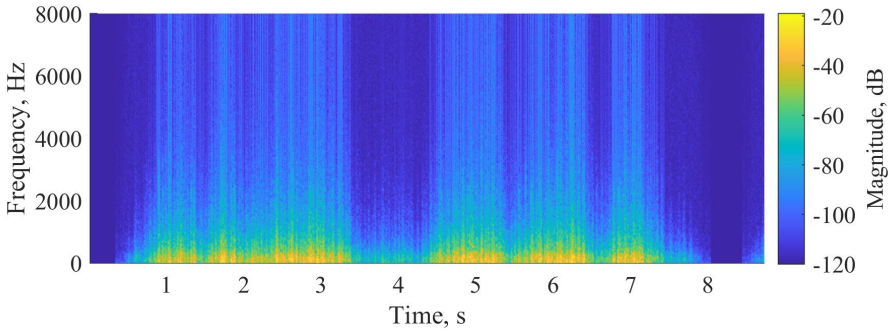

Figure 6.3: Sample of the wind signal used for gusty wind simulation

6.2 Results

All algorithms are evaluated using the PESQ speech-intelligibility measure ([2]). For the first simulation, the speech signal and the wind signal were added to form a single mixed audio file (results - in Fig. 6.4 and Fig. 6.5). The PESQ improvement was measured for different SNR combinations, ranging from $-10dB$ to $30dB$ - shown in Fig. 6.8. For the second simulation, we use simulated wind noise, changing the type of the wind to gusty wind instead of continuous wind. The results are shown in Fig. 6.6 and Fig. 6.7. The PESQ results are shown in Fig. 6.9.

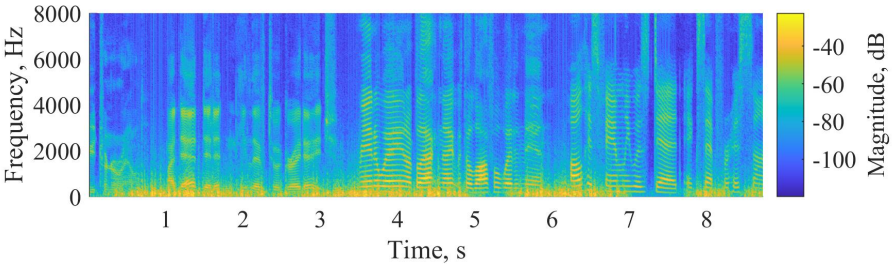

Figure 6.4: Sample of the input signal for continuous wind simulation

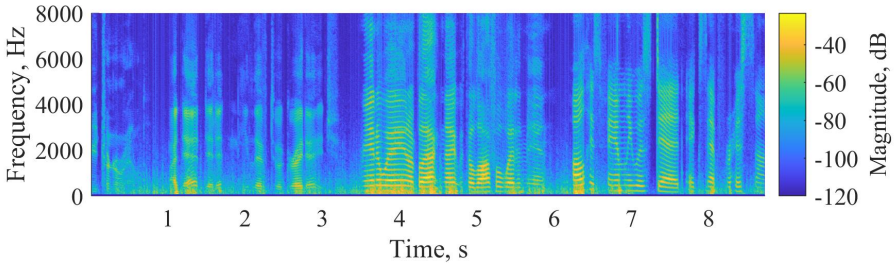

Figure 6.5: Sample of the output signal for continuous wind simulation

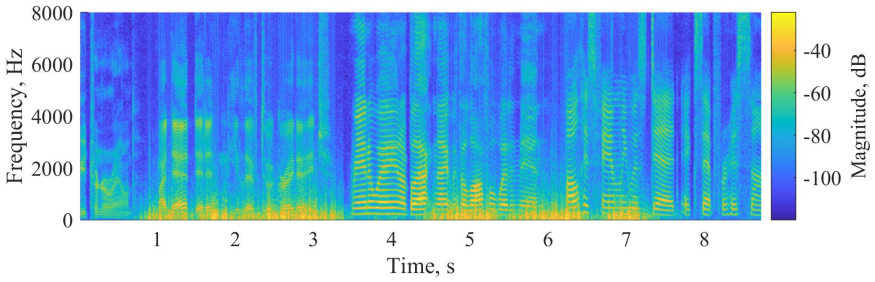

Figure 6.6: Sample of the input signal for gusty wind simulation

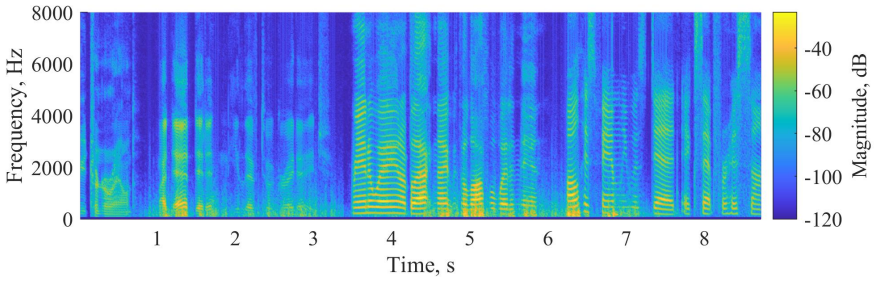

Figure 6.7: Sample of the output signal for gusty wind simulation

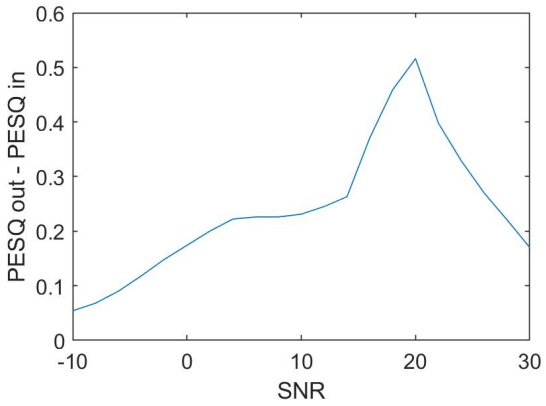

Figure 6.8: PESQ measure for continuous wind

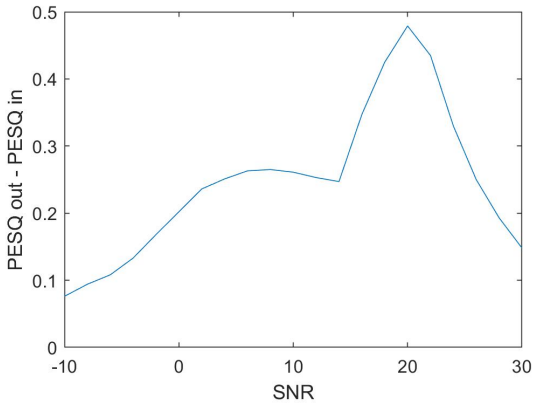

Figure 6.9: PESQ measure for gusty wind

The results for continuous wind and gusts of wind appear to be similar to each other. For a SNR of 20, the best improvement in PESQ was measured. However, some sort of improvement can be see for every measured SNR, the lowest being for the worst SNR - as expected.

A possible reason for the decline for SNRs bigger than 20 could be misdetection of wind due to high SNR. In such cases, most of the mixed segments would be classified as wind-absent regions, and the signal would remain relatively untouched. For low SNRs, a relatively small improvement of PESQ might happen due to the wind noise being too dominant, thus classifying mixed segments as pure wind - reducing wind and speech altogether, i.e. maintaining the same SNR.

6.3 Conclusions

In this project, a system was proposed which exploits the characteristics of the wind noise and human voice to detect and reduce wind noise in noisy single microphone speech signals. To conclude, we did manage to clean a speech signal with wind noise to some extent. We have encountered a trade-off between having a more continuous speech, and reducing the noise of the wind. We have chosen to give up some of the reduction for the signal to sound smoother. With that being said, our algorithm is far from perfect, and we would like to suggest some ideas for future improvement.

First, we believe that the way we have chosen our parameters is not the optimal one, and that could be solved by using machine learning algorithms, and more big-data analysis.

In addition, we have made a couple of questionable assumptions we would like to further inspect. The first one - the characteristics of the wind noise. Wind indeed looks like a decaying exponential function, but we believe that further research of wind noise could give us a more accurate model - thus improving the estimation of its PSD. The second one - the structure of the human voice. We have assumed that the PSD of the human voice could be reconstructed using a single a pitch harmony, and multiplications of it. We believe this model is merely a rough estimate, since the human voice is far more complicated than that. Further research of the human voice could help us with estimating the PSD of the wind noise, as explained in chapter 5.

Bibliography

[1] Christian Hofmann, Tobias Wolff, Markus Buck, Tim Haulick, and Walter Kellermann. A morphological approach to single-channel wind-noise suppression. In *IWAENC 2012; International Workshop on Acoustic Signal Enhancement*, pages 1–4. VDE, 2012.

[2] O ITU. Series p: Telephone transmission quality, telephone installations, local line networks-subjective video quality assessment methods for multimedia applications, 2008.

[3] Christoph Nelke, Peter Jax, and Peter Vary. Wind noise detection: Signal processing concepts for speech communication. *Energy*, 60(40):20, 2016.

[4] Christoph M Nelke and Peter Vary. Wind noise short term power spectrum estimation using pitch adaptive inverse binary masks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5068–5072. IEEE, 2015.

[5] Christoph Matthias Nelke, Navin Chatlani, Christophe Beaugeant, and Peter Vary. Single microphone wind noise psd estimation using signal centroids. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7063–7067. IEEE, 2014.

[6] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[7] Anthony D Rhodes. Real-time wind noise detection and suppression with neural-based signal reconstruction for mult-channel, low-power devices. *arXiv preprint arXiv:1710.00082*, 2017.

This project is dedicated to our families.
We would like to thank, first and foremost, our mentor Dr. Ofer Schwartz for helping us through each and every step along the way. We would like to thank Prof. Sharon Gannot and the faculty of engineering in Bar-Ilan university for giving us the opportunity to explore such themes and gather a lot of important knowledge. We would like to thank the scholars who explored this subject before us and have given us a good basis for this project.

Appendices

Appendix A

Code

A.1 Main

```
[x, fs] = audioread('input.wav');   % load an audio file

% ALGORITHM %
%***********%

% ANALYSIS PARAMETERS %
wlen = 512;                          % window length
hop = wlen/4;                         % hop size
nfft = wlen;                          % number of fft points
xlen = length(x);
win=hamming(wlen);

% ANALYSIS %
[X,t,f] = stft(x, win, hop, nfft, fs);
% TIME AND FREQUENCY LENGTH %
M=wlen/2+1; %Frequency length
L=length(X(1,:)); %Time length

% INITIATING ARRAYS %
rvv=zeros(M,1);
rxx=zeros(M,1);
rxx_mat=zeros(M,L);
rvv_mat=zeros(M,L);
b_m=8*ones(10,1); %Array of Mu_0 for each time stamp
X_est=zeros(M,L); %Estimated clean signal
b_m_arr=zeros(L,1);
ind_arr=zeros(3,1);
mean_arr=zeros(3,1);
mean_wind=zeros(3,1);
mean_voice=zeros(3,1);
cov_voice=zeros(3,3);
cov_wind=zeros(3,3);
cov_mat=zeros(3,3);
zcr_=0;
nsf_=0;
ssc_1=0;

% SETTING OF PARAMETERS %
G_min=0.01;
```

```
G_max=0.1;

Local_thres_low_freq = 0.01;
Local_thres_high_freq = 0.1;
mid_hz_Thres = 300;
mid_point = 1/(mid_hz_Thres/(16000/2)*(512/2));
t_exp = 0:1:512/2;
G_max_exp= (Local_thres_low_freq - Local_thres_high_freq)*exp(-mid_point*t_exp)...
    + Local_thres_high_freq;

Local_thres_low_freq = 0.01;
Local_thres_high_freq = 0.03;
mid_hz_Thres = 300;
mid_point = 1/(mid_hz_Thres/(16000/2)*(512/2));
t_exp = 0:1:512/2;
G_min_exp= (Local_thres_low_freq - Local_thres_high_freq)*exp(-mid_point*t_exp)...
    + Local_thres_high_freq;
a_min=0.3;
a_max=0.9;
low=80;
high=450;
m_delta=round(60/(fs/2)*M);
m_l=round(100/(fs/2)*M);
m_h=round(2000/(fs/2)*M);
f_bins=(1:M);
f_max_wind=(1000/(fs/2)*M);
% pitch detector
ff=80:10:250; % f0 range
tt=1./ff*fs;
pitch_thresh = 0.5;
bins = 10:200;

for n=1:length(ff)
    F_t(:,n) = exp(sqrt(-1)*(bins-1)*2*pi/wlen*tt(n));
end


load('model_wind.mat')
load('model_voice.mat')



%%%%%% SIGNAL PROCESSING %
time=1;
for l=1:L-1
    time=time+128;
    %%%%%% PSD OF X %
    a_rxx=0.95;
    help = abs(X(:,l)).^2;
    for m=1:M
        if help(m) > rxx(m)
            rxx(m)=(a_rxx)*rxx(m) + (1-a_rxx)*help(m); % Smoothing
        else
            rxx(m)=0.5*rxx(m) + (1-0.5)*help(m); % Smoothing
        end
        rxx_mat(:,l) = rxx;
```

```
    end

    %%%%%% WIND DETECTION %
    xw = x(1+l*hop : wlen+l*hop).*win;
    %%%%%%
    zcr_ = 0.95*zcr_ + 0.05*zcr(xw);
    zcr_out(1+l*hop : wlen+l*hop) = zcr_;
    %%%%%%
    nsf_ = nsf(  log(rxx(floor(1:f_max_wind))) , f_bins(floor(1:f_max_wind))'); % Negative Slope Fit
    nsf_ = 1-nsf_;
    if nsf_>0
        nsf_out(1+l*hop : wlen+l*hop) = nsf_;
    end
    %%%%%%
    ssc_1 = ssc(rxx,m_l,m_h); % Spectral Centroid
    if ssc_1>0
        ssc_out(1+l*hop : wlen+l*hop) = ssc_1;
    end


    %%%%%% DEFUSED DETECTION %
    ind_arr = [ zcr_; nsf_  ;ssc_1] ;

    decision = Decision(ind_arr, mean_voice, mean_wind, cov_voice_inv, cov_wind_inv);
    decision_out(1+l*hop : wlen+l*hop) = decision;

    %%%%%% PITCH ESTIMATION %

    [pitch] = pitch_est(rxx, F_t, pitch_thresh, bins, wlen, fs, ff);
    pitch_out(1+l*hop : wlen+l*hop) = pitch;

    %%%%%% NOISE INTERPOLATION (Binary Mask Algorithm in Pitch Freq.)
    if (decision > low) && (pitch > 0) && (decision < high)
        N_c = Noise_interpolation(rxx, pitch, m_delta);
    else
        if (decision >= high)
            N_c = 0 ;
        else
            N_c = rxx;
        end
    end

    %%%%%% WIND NOISE PSD ESTIMATION %

    rvv = N_c;
    rvv_mat(:,l) = rvv;

    %%%%%% WIENER FILTER %
    G = calc(decision,low,high,G_min_exp,G_max_exp); % Wiener filter wind indicator due to ssc

    if l==1
        [X_est(:,l)] = Wiener(X(:,l), rxx, rvv, 0, 0.95, G');
    else
        [X_est(:,l)] = Wiener(X(:,l), rxx, rvv, X_est(:,l-1), 0.95, G');
    end
```

```
    %%%%%% DC filtering %
    X_est(1:3,l)=0;
end

% SYNTHESIS %
x_est=istft(X_est, win, win, hop, nfft, fs);

% END OF ALGORITHM %
```

A.2 Functions

A.2.1 STFT

```
function [STFT, f, t] = stft(x, win, hop, nfft, fs)
% function: [STFT, f, t] = stft(x, win, hop, nfft, fs)
%
% Input:
% x - signal in the time domain
% win - analysis window function
% hop - hop size
% nfft - number of FFT points
% fs - sampling frequency, Hz
%
% Output:
% STFT - STFT-matrix (only unique points, time
%        across columns, frequency across rows)
% f - frequency vector, Hz
% t - time vector, s
% representation of the signal as column-vector
x = x(:);
% determination of the signal length
xlen = length(x);
% determination of the window length
wlen = length(win);
% stft matrix size estimation and preallocation
NUP = ceil((1+nfft)/2);     % calculate the number of unique fft points
L = 1+fix((xlen-wlen)/hop); % calculate the number of signal frames
STFT = zeros(NUP, L);       % preallocate the stft matrix
% STFT (via time-localized FFT)
for l = 0:L-1
    % windowing
    xw = x(1+l*hop : wlen+l*hop).*win;

    % FFT
    X = fft(xw, nfft);

    % update of the stft matrix
    STFT(:, 1+l) = X(1:NUP);
end
% calculation of the time and frequency vectors
t = (wlen/2:hop:wlen/2+(L-1)*hop)/fs;
f = (0:NUP-1)*fs/nfft;
end
```

A.2.2 ISTFT

```
function x = istft(STFT, awin, swin, hop, nfft, fs)
% function: [x, t] = istft(STFT, awin, swin, hop, nfft, fs)
%
% Input:
% stft - STFT-matrix (only unique points, time
%        across columns, frequency across rows)
% awin - analysis window function
% swin - synthesis window function
% hop - hop size
% nfft - number of FFT points
% fs - sampling frequency, Hz
%
% Output:
% x - signal in the time domain
% signal length estimation and preallocation
L = size(STFT, 2);          % determine the number of signal frames
wlen = length(swin);        % determine the length of the synthesis window
xlen = wlen + (L-1)*hop;    % estimate the length of the signal vector
x = zeros(1, xlen);         % preallocate the signal vector
% reconstruction of the whole spectrum
if rem(nfft, 2)
    % odd nfft excludes Nyquist point
    X = [STFT; conj(flipud(STFT(2:end, :)))];
else
    % even nfft includes Nyquist point
    X = [STFT; conj(flipud(STFT(2:end-1, :)))];
end
% columnwise IFFT on the STFT-matrix
xw = real(ifft(X));
xw = xw(1:wlen,:);
% Weighted-OLA
for l = 1:L
    x(1+(l-1)*hop : wlen+(l-1)*hop) = x(1+(l-1)*hop : wlen+(l-1)*hop) + ...
                                      (xw(:, l).*swin)';
end
% scaling of the signal
W0 = sum(awin.*swin);
x = x.*hop/W0;
end
```

A.2.3 ZCR

```
function zcr=zcr(xw)
% Zero Crossing Rate %
L=length(xw);
zcr=0;
for l=2:L-1
    if (xw(l)*xw(l-1)<0)
        zcr=zcr+1;
    end
end
zcr=zcr/(L-2);
end
```

A.2.4 NSF

```
function [ind]=nsf(X,f)
M_nsf=[ones(length(f),1),f];
I=eye(2);
A_nsf=I/((M_nsf.')*M_nsf);
a_opt_nsf=A_nsf*(M_nsf.')*X;
X_est_nsf=M_nsf*a_opt_nsf;
%%%

if a_opt_nsf(2) < 0
    ind = 1 - (norm(X-X_est_nsf)^2 / norm(X)^2);
else
    ind = 0;
end

end
```

A.2.5 SSC

```
function ssc=ssc(X,m_l,m_h)
% Spectral Centroid Function %
X=X(m_l:m_h);
ssc=[1:m_h-m_l+1]*X/(sum(X)+ 10^-8);
 ssc = ssc / (m_h-m_l);
end
```

A.2.6 Decision

```
function decision=Decision(ind_arr, mean_voice, mean_wind, cov_voice_inv, cov_wind_inv )
%decision=0; %there is no wind, H0
help_wind = log(det(cov_wind_inv));
help_voice = log(det(cov_voice_inv));
f_x_H0 = -0.5*(ind_arr-mean_wind)'*(cov_wind_inv)*(ind_arr-mean_wind) + 0.5*log(det(cov_wind_inv)); %
f_x_H1 = -0.5*(ind_arr-mean_voice)'*(cov_voice_inv)*(ind_arr-mean_voice) + 0.5*log(det(cov_voice_inv)
decision = f_x_H1 - f_x_H0;
end
```

A.2.7 Pitch est

```
function [pitch] = pitch_est(X,F_t,pitch_thresh_1,bins,window,fs,f)
%%% pitch based VAD
help = X(bins); %abs(Input_mics(1,t)).^2;%
Likelihood = real(F_t'*help ) ;
[ma, pos] = max(Likelihood);
pitch=0;
if ma > pitch_thresh_1 * sum(help)
    pitch = round(f(pos)*window/fs);
end
```

A.2.8 Noise interpolation

```
function [N_est] = Noise_interpolation(rxx,pitch, m_delta)

% BINARY MASK %
% Linear interpolation between 2 points in the beginning and end of the
% pitch harmony region.
N_est=rxx;
```

```
a_1=1;
num_jumps=floor(length(rxx)/pitch);
for i=1:num_jumps
        if ((i*pitch-m_delta)>1)
            a_1=i*pitch-m_delta;
        end
        b_1=min([i*pitch+m_delta length(N_est)]);
        N_est(a_1:b_1)=LinInterp(N_est(a_1),N_est(b_1),b_1-a_1+1);
end
end
```

A.2.9 calc

```
function [ind]=calc(decision,low,high,dec_min,dec_max)
% Only noise region, we feed rvv with the new noise
if (decision<=low)
    ind=dec_min;
end
% Noise and speech region
if (decision > low && decision < high)
    ind=((dec_min)*(high-decision) + dec_max*(decision-low))/(high-low);
end
% Only speech region, we do not feed rvv with much of the new information
if (decision>=high)
    ind=dec_max;
end
end
```

A.2.10 Wiener

```
function [GX_next]=Wiener(X, rxx, rvv, GX_prev, a,G_min)
% Wiener Filter Function %
% Smoothing of speech estimation (decision-directed)
rss = a*abs(GX_prev).^2+(1-a)*max(rxx-rvv,0);
% Wiener Filter
G = rss./(rss+rvv);
GX_next = max(G,G_min).*X;
end
```