IN4182: Digital Audio and Speech Processing

Report

Erik Hagenaars
4272404

Teun de Smalen
4321014

TODO LIST

K10 en Kinf fixen (onderkant valt er vanaf) 5

I. Introduction

The use of microphones and audio devices are becoming more relevant every year. Speech audio is used to communicate humans and recently often with devices as well. Hearing loss and noisy spaces are an increasing problem which makes it harder to communicate. It is therefor important to come up with techniques to enhance the desired speech signal from a noisy environment. This comes with many challenges including correlating noises, dynamic noise energies and interference noise. This report discusses the work done on the mini-project for the course "IN4182 - Digital Audio and Speech Processing" in which a single microphone speech enhancement system is designed.

The first objective is to create an architecture of the system, which is discussed in Section II. From this, five subsystems are designed and implemented in the Sections III to VII. After designing the system, its performance will be tested and concluded in Section IX.

As a bonus, a multi microphone system is designed with the implementation of beamformers in Section VIII.

II. System

A. Signal model

Before the architecture of the system can be designed, the signal model and assumptions are made. For the signal model, Additive White Gaussian Noise (AWGN) is expected. If Y is defined as the signal with AWGN, S the desired source signal and N the noise, the model can be expressed as shown in Eq. 1 and Eq. 2. In which the time domain and frequency domain expressions are shown.

$$Y_t[n] = S_t[n] + N_t[n] \quad \text{(time domain)} \tag{1}$$
$$Y_k[l] = S_k[l] + N_k[l] \quad \text{(frequency domain)} \tag{2}$$

To simplify the model more, some assumptions are made. The first assumption is that the source signal and noise are uncorrelated (Eq. 3). This allows for the autocorrelation of the received signal to be simplified. Since the source and noise are uncorrelated, the autocorrelation of the received signal can be expressed by the addition of the autocorrelation of the signal and the noise (Eq. 4). The second assumption is that the speech signal is wide-sense stationary (WSS) in small frames 5). Since a speech signal can be seen as a periodic signal or noise, this assumption holds in theory. In practise, speech is not stationary but the performance of the enhancement system is sufficient.

$$R_{S_t N_t}(n, m) = 0 \quad \text{(uncorrelated)} \tag{3}$$
$$R_{Y_t Y_t}(n, m) = R_{S_t S_t}(n, m) - R_{N_t N_t}(n, m) \tag{4}$$
$$R_{Y_t Y_t}(n, m) = R_{Y_t Y_t}(m - n) \quad \text{(wide-sense stationary)} \tag{5}$$

An important property of audio is the power spectrum density (PSD). The PSD is defined as the Fourier Transform (FT) of the autocorrelation (Eq. 6). And with the assumption of uncorrelated noie and source, this can be expressed as the PSD of the signal and noise added as in Eq. 7. Since the frames are of finite length, an estimation of the PSD needs to be made. The estimation of Eq. 8 is called the periodogram of the signal. To enhance this estimation, Bartlett's method can be used shown in Eq. 9.

$$P_{YY,k} = \lim_{L\to\infty} \sum_{m=-L/2}^{L/2} R_{Y_t Y_t}(m) e^{-j2\pi \frac{km}{K}} \tag{6}$$

$$= P_{SS,k} + P_{NN,k} \tag{7}$$

$$\hat{P}^{P}_{YY,k}(l) = \frac{1}{L} |Y_k(l)|^2 \tag{8}$$

$$\hat{P}^{B}_{YY,k}(l) = \frac{1}{M} \sum_{m=l-M+1}^{l} \hat{P}^{P}_{YY,k}(m) \tag{9}$$

B. System overview

From the signal model, a few components of the system can be derived. Firstly, to hold the WSS assumption, the signal sequence needs to be split in multiple segments. This process will be expressed as Framing. Since the PSD is an important property of the signal, it need to be converted to the frequency domain. A problem is that the FT can cause problems in the sidelobes of the frames due to Gibb's phenomenom. To avoid this, the framing function needs to window this signal frame to suppress the edges of the frame. The frames will become overlapping. At the end of the system, the time domain representation of the signal will be recovered using the inverse FT and the frames will be merged using the overlapp add method where a deframing window is used. To estimate the source, some other properties need to be estimated from the received signal. These important properties include the estimated noise PSD, the estimated source PSD (by estimating the SNR) and if the source is present. With these properties, a gain filter can be created for the received signal which should suppress the noise and interference. The resulting system design is shown in Fig. 1.

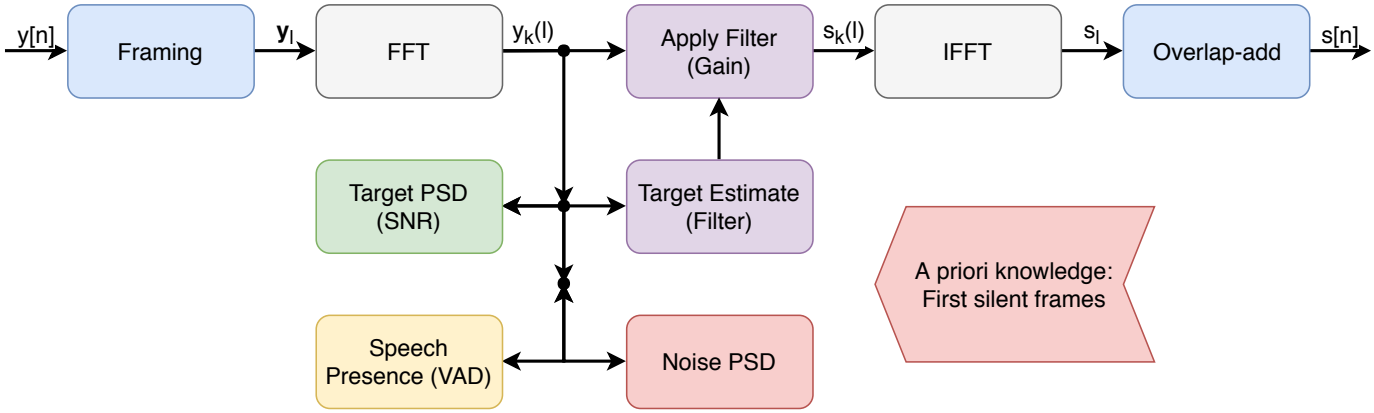

Fig. 1: Overview of the system.

This system is focused on a single microphone setup. With a multiple microphone setup, the speech enhancement can be improved upon. Direction estimation and spatial filtering with beamformers can be implemented to filter the unwanted interference and noise from certain directions. This subsystem can be placed before the single microphone speech enhancement system.

The system's functions will be divided into six Sections. The framing and overlap add function will be implemented in Section III. The noise PSD estimation and the source PSD estimation are discussed in Section IV and Section V respectively. The voice activity detection will be discussed in Section VI. With all this information, the target estimation where the filters are designed is discussed in Section VII. And lastly, the spatial filtering will be discussed in Section VIII.

III. Framing & Overlap Add

The first step, as described in Fig. 1, is the framing of the audio file. This is done according to Eq. 10. Where l is the frame index (the l-th frame), n is sample number, R is the hoplength. $w[n]$ is the window used to smoothen the signal in such a way that the signal does not become discontinuous and cause wrong sidelobes wen applying the FT.

$$y_l[n] = y[n + lR]w[n], \quad n = 0, \ldots, N-1 \tag{10}$$

The last step of the system is the Overlap Add-block. The windowing is removed after which the samples are added back together to one file.

$$y[n] = \sum_{l=1}^{k} y_l[n]/w[n] \tag{11}$$

There are various windows that are suited for framing an audio signal. The standard Hamming and Hann window and the square-root Hann window used in a paper by Hendriks[1] was evaluated. When overlap adding all the frames without applying any changes, the signal should be recovered without any trouble. The overlap added windows should then be equal to an ones vector. Using different values for the overlap percentage, the corresponding overlap add vector is generated and compared to an all-ones vector. An important variable for the framing function is the overlap percentage. The results in Fig. 2 show the negative mse of these two vectors. The peak value is around 70% for the square-root Hann.

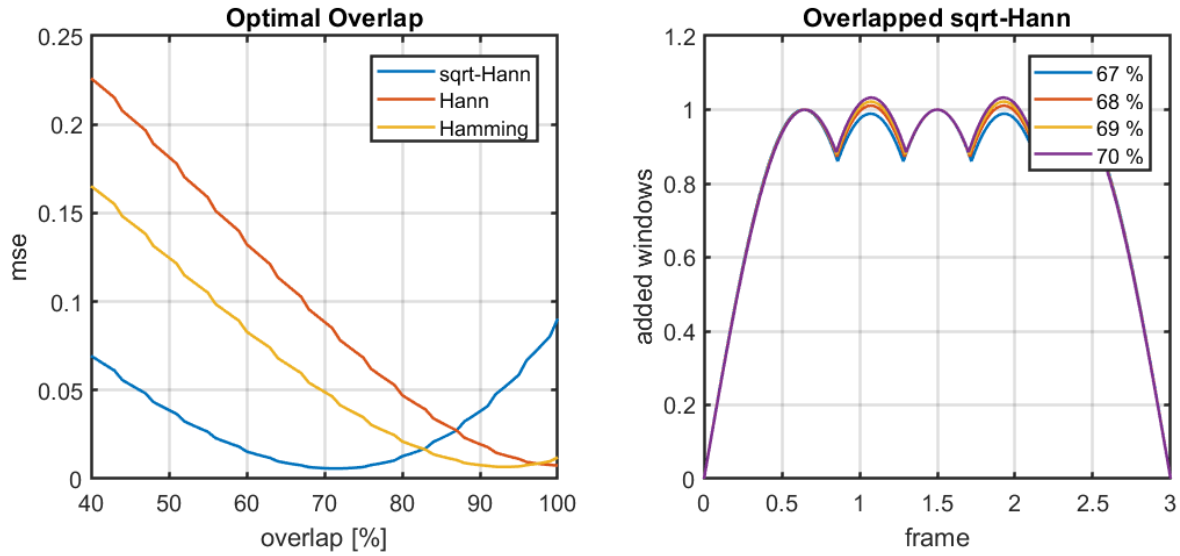

Fig. 2: Optimal overlap percentage for the square-root Hann window.

IV. Noise PSD Estimation

For the noise estimation, it is considered that speech and noise are independent and uncorrelated. From this Eq. 12 and 13 can be derived. Whether H_0 or H_1 corresponds to a particular timeframe is determined by the Voice Activity Detector in Section VI.

$$H_0 : Y_K(l) = N_k(l) \quad \text{(speech absence)} \tag{12}$$
$$H_1 : Y_K(l) = S_k(l) + N_k(l) \quad \text{(speech presence)} \tag{13}$$

A. Voice Activity Dectector

The method used to estimate the noise PSD is based on Voice Activity Detector. This Noise PSD estimation is given as Eq. 14. Where α is the smoothing constant. As described in [1], this method works well for noise with low variation in time.

$$\sigma^2_{N,k}(l) = \begin{cases} \alpha\hat{\sigma}^2_{N,k}(l-1) + (1-\alpha)\left|y_k(l)\right|^2 & \text{when } H_0(l) \\ \hat{\sigma}^2_{N,k}(l-1) & \text{when } H_1(l) \end{cases} \tag{14}$$

In Fig. 3, the results are shown. From the figure, the noise is canceled for a great part. Especially in the speech absence regions. This is Because of the Wiener filter implementation in combination with the excelent noise estimate when speech is not present. The speech presence regions are uncertain and can have significant errors. This uncertainty is expected to be due to non-optimal variables and reliability on the VAD. When a system becomes more reliable on other system, its robustness could decline when the subsystems are not accurate.

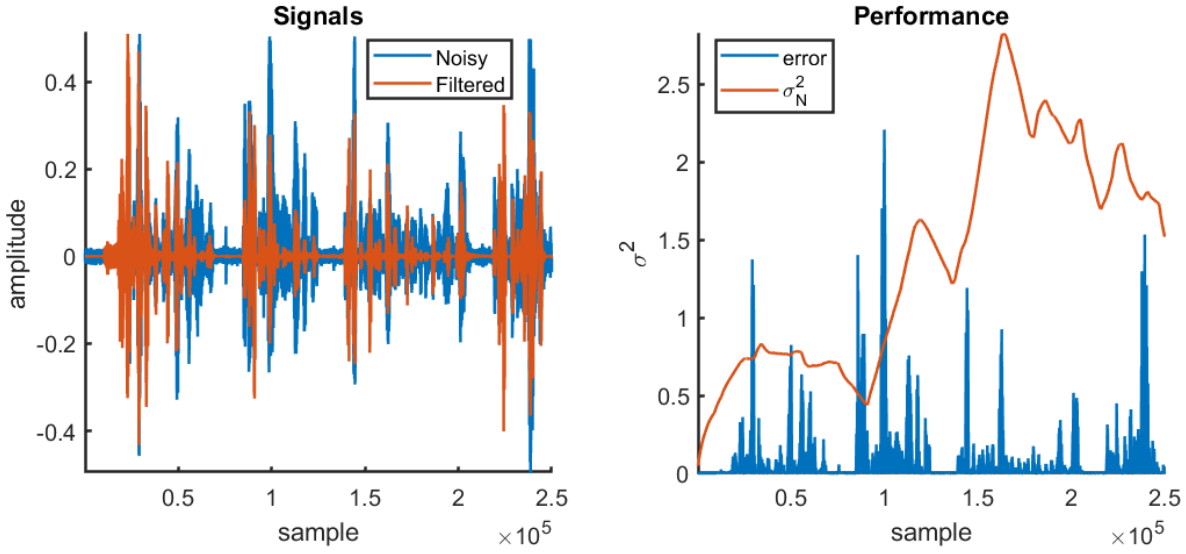

Fig. 3: Results of the VAD based approach for estimating the noise PSD.

B. Minimum Statistics Method

The Minimum Statistics Method does not depend on the Voice Activity Detector and only depends on previous time frames of y_k as can be seen in Equation 16 and 17. Because using only the minimum value would be a underestimate of $E(|N|^2)$. A biasing factor, B_{min} is added. This biasing factor used is the mean of the 50 previous values of $Q_k(l)$, giving K=50.

$$Q_k(l) = \alpha_k(l)Q_k(l-1) + (1-\alpha_k(l))\,|y_k(l)|^2 \tag{15}$$

$$\mathbf{Q} = \{Q_k(l-L+1) \quad \dots \quad Q_k(l)\} \tag{16}$$

$$\hat{\sigma}^2_{N,k}(l) = B_{min}Q_{min} \tag{17}$$

$$\tag{18}$$

In Fig. 4, the results are shown. Here it shows that the noise is canceled very well. When more frames pass, the error increases in voiced areas. This could be due to the bias factor which was added or other mis-estimated variables. It seems to perform better than the VAD based approach in general. A big positive for this method is its independence.

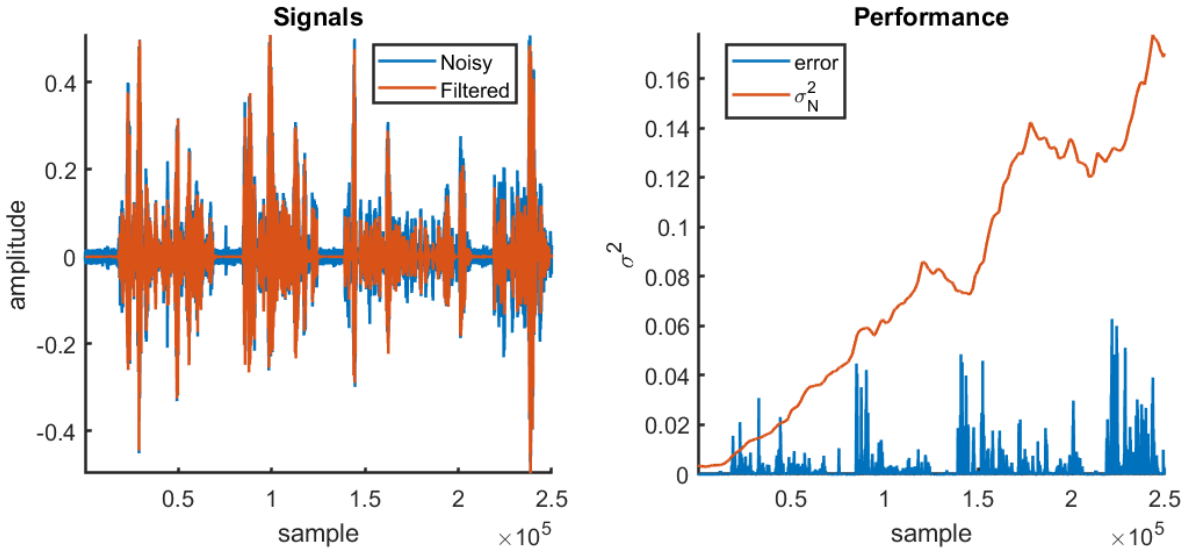

Fig. 4: Results of the Minimal Statistics based approach for estimating the noise PSD (K=50).

An important variable is the number of previous frames used. To test the optimal number of previous frames, tests are made for $K = 10$ and $K = Inf$. The results of these tests are shown in Fig. 5 and 6. For a lower amount of previous frames, the error increases. Thus by increasing the amount of previous frames used, the error should decrease. This is shown when all previous frames are used, giving K is infinite. Comparing this to the two tries before indeed shows a lower error present. While compared to K=10, there is a 50% execution-time increase.

K10 en Kinf fixen (onderkant valt er vanaf)

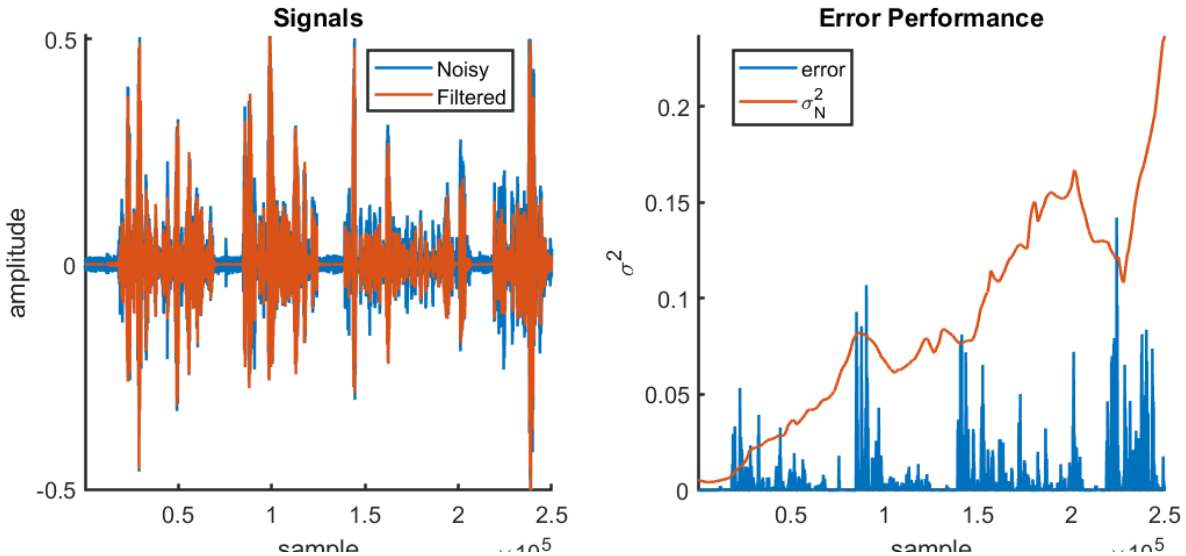

Fig. 5: Results of the Minimal Statistics based approach using K=10.

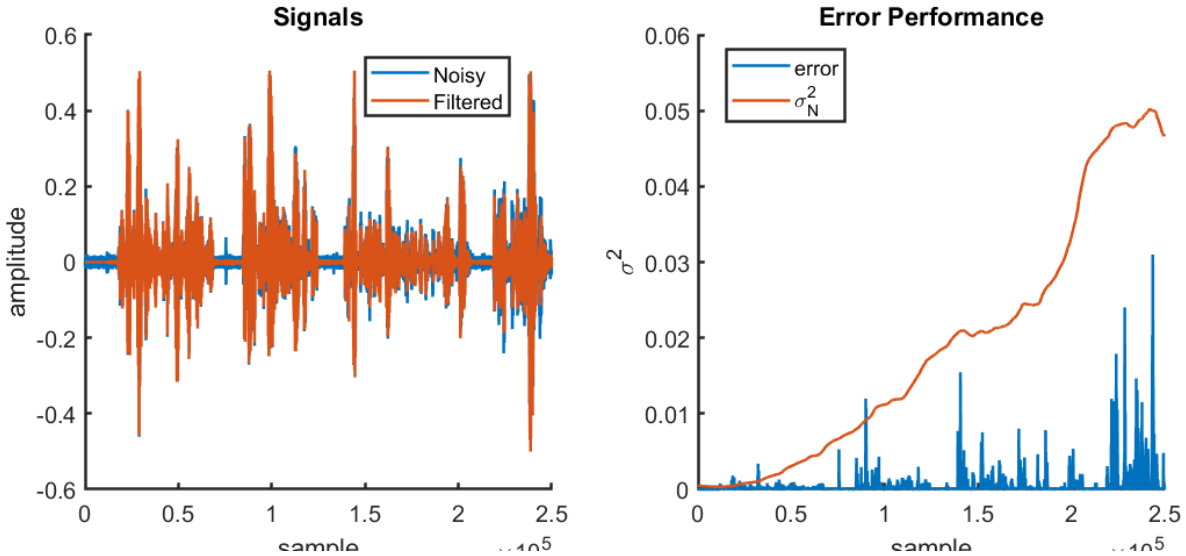

Fig. 6: Results of the Minimal Statistics based approach using K=Inf.

C. MMSE Method

One of the more extensive methods discussed during the course is the MMSE Method. Equation 19 looks similar to Equation 14 when speech is not present. Instead of using the PSD of the input signal $y_k(l)$, the noise its expected value is determined for that specific time-frame. This can be done as in Equation 20.

$$\widehat{\sigma_N^2}(l) = \alpha\widehat{\sigma_N^2}(l-1) + (1-\alpha)E\left[|N_k(l)|^2 |y_k(l)|\right] \tag{19}$$

$$E\left[|N_k(l)|^2 |y_k(l)|\right] = P\left(H_{0,k}(l)|y_k(l)\right) E\left[|N_k(l)|^2 |y_k(l)|, H_{0,k}\right] + P\left(H_{1,k}(l)|y_k(l)\right) E\left[|N_k(l)|^2 |y_k(l)|, H_{1,k}\right] \tag{20}$$

There are some unknowns present. The change that speech is absent can be rewritten as in Equation 21. The expected values can be given as Formulas 22 and 23.

$$P\left(H_{0,k}(l)|y_k(l)\right) = 1 - P\left(H_{1,k}(l)|y_k(l)\right) \tag{21}$$

$$E\left[|N_k(l)|^2 |y_k(l)|, H_{0,k}\right] = \left|y_k(l)^2\right| \tag{22}$$

$$E\left[|N_k(l)|^2 |y_k(l)|, H_{1,k}\right] = \widehat{\sigma_N^2}(l-1) \tag{23}$$

This leaves only one function to be determined as which can be further broken down as in Equation 24. Of which the two elements are given by Equations 25 and 26.

$$P\left(H_{1,k}(l)|y_k(l)\right) = \frac{P\left(H_{1,k}(l)\right)) p_{Y|H_1}}{P\left(H_{1,k}(l)\right)) p_{Y|H_1} + P\left(H_{0,k}(l)\right)) p_{Y|H_0}} \tag{24}$$

$$p_{Y|H_0} = \frac{1}{\widehat{\sigma_N^2}\pi} \exp\left(-\frac{|y^2|}{\widehat{\sigma_N^2}}\right) \tag{25}$$

$$p_{Y|H_0} = \frac{1}{\widehat{\sigma_N^2}(1+\xi_{H_1})\pi} \exp\left(-\frac{|y^2|}{\widehat{\sigma_N^2}(1+\xi_{H_1})}\right) \tag{26}$$

However, ξ_{H_1} is unknown and there is no sufficient a priori knowledge present to give an estimation of this variable. Which makes this method unfeasible to use.

V. SNR Estimation

The SNR estimation is used for a few subsystems. The Voice Ativity Detection in Section VI and the Target Estimation for the Wiener filter in Section VII. There are two main approaches to estimate the SNR, these are discussed in this Section. First the definition of the SNR is written down in Eq. 27. The SNR is the power of the signal divided by the power of the noise. Often, this is expressed in decibel.

$$\xi = \frac{\sigma_{S,k}(l)^2}{\sigma_{N,k}(l)^2} = \frac{P_{SS,k}}{P_{NN,k}} = \frac{E\left\{|S_k(l)|^2\right\}}{E\left\{|N_k(l)|^2\right\}} \tag{27}$$

A. Maximum Likelihood

The first estimation approach is based on the maximum likelihood. Here, the signal power is estimated by the Bartlett estimate of the input power. The expected value of the noise power is divided by the order of the Bartlett estimate to use as the denominator.

$$\xi_k(l) = \frac{E\left\{|Y_k(l)|^2\right\}}{E\left\{|N_k(l)|^2\right\}} - 1 \tag{28}$$

$$= \frac{\hat{P}^B_{YY,k}(l)}{\frac{1}{L}E\left\{|N_k(l)|^2\right\}} \tag{29}$$

The first testresults of the estimator can be seen in Fig. 7. The plot does not include the SNR lower than 0. The SNR estimates are not accurate since the SNR in the test was set at 20dB and the mean of the SNR estimates was 12dB. During speechless periods, it dit give a very small SNR estimate. This could be used as a good indicator for the VAD system.

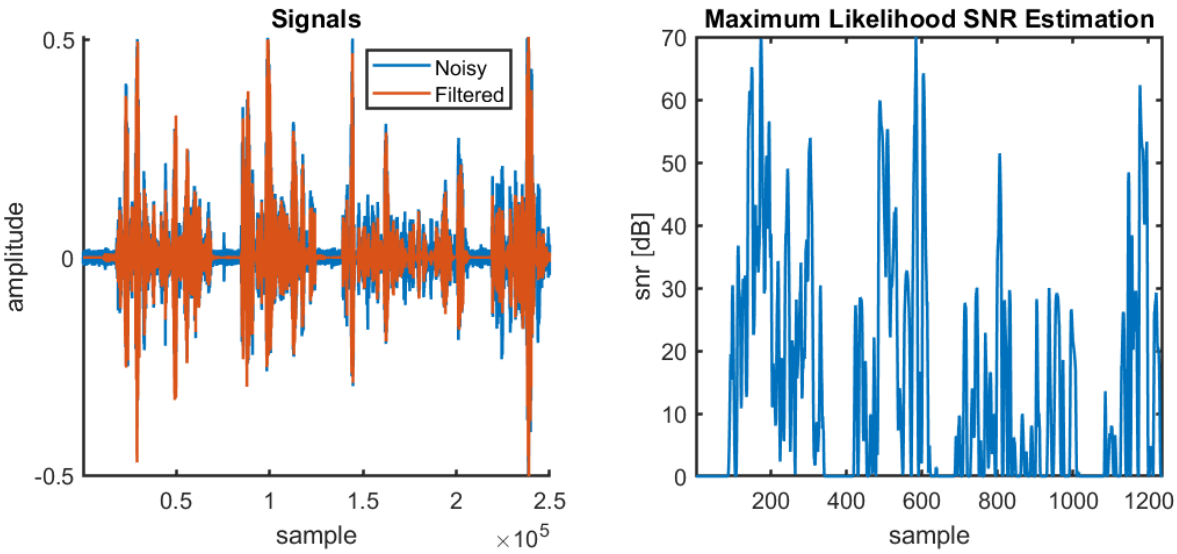

Fig. 7: SNR Estimation using the ML approach.

B. Decision Directed

The second estimation approach is more practical. The decision directed approach is making a current SNR estimation with the estimation from the previous frame. A smoothing variable α is used. This can be seen in Eq. 30. Assumed is that the previous SNR estimate is known. The current SNR estimate is using the equations found in Eq. 31 and 32.

$$\xi_k(l) = \alpha \frac{E\left\{|S_k(l)|^2\right\}}{E\left\{|N_k(l)|^2\right\}} + (1-\alpha)\left(\frac{E\left\{|Y_k(l)|^2\right\}}{E\left\{|N_k(l)|^2\right\}} - 1\right) \tag{30}$$

$$|S_k(l)|^2 = \left|\hat{S}_k(l-1)\right|^2 \tag{31}$$

$$\frac{E\left\{|Y_k(l)|^2\right\}}{E\left\{|N_k(l)|^2\right\}} - 1 = \max\left[\left(\frac{|Y_k(l)|^2}{E\left\{|N_k(l)|^2\right\}} - 1, 0\right)\right] \tag{32}$$

The results of the implemented estimator are shown in Fig. 8. In comparison to the ML approach, the estimation is much smoother as expected. The areas of speech are more defined. The average SNR on all frames was close to the SNR set on the variables of the model.

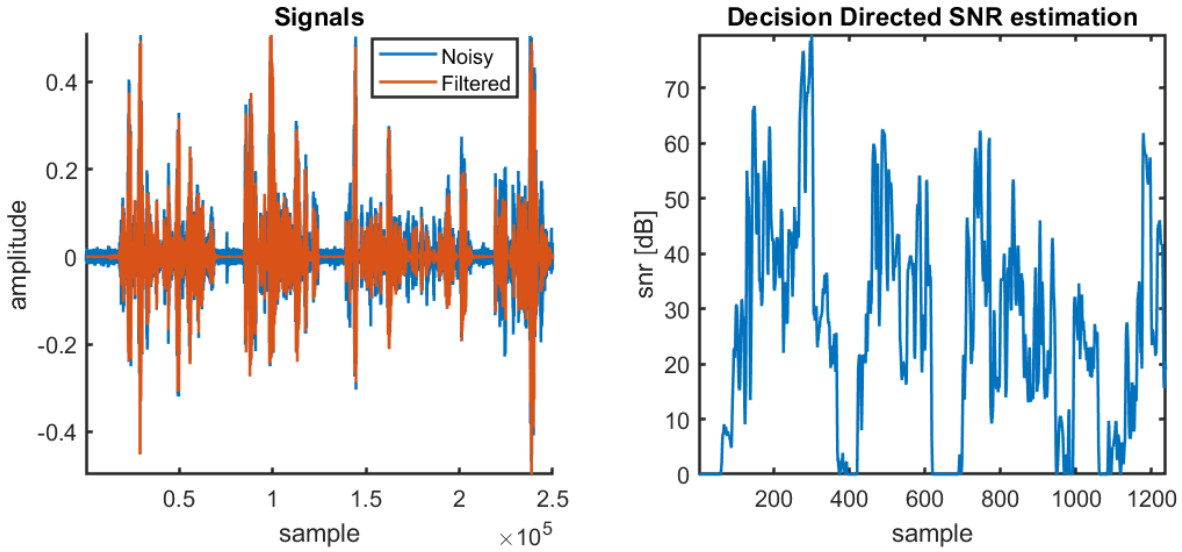

Fig. 8: SNR Estimation using the DD approach.

VI. Voice Activity Detection

For the noise PSD estimation in Section IV and the SNR estimation in Section V, the pressence of speech is needed to increase the performance of the estimators. The Voice Activity Detector will evaluate the frame and give a speechflag wether speech is present or not.

This can be done by looking at the average energy of the frame. In the case of speech present and a SNR bigger than 0, this should be higher than in the case of noise only. To get a linear scaling with respect to the SNR, the log-energy is calculated. We then can express the energy difference as stated in Eq. 35 and Eq. 36. This likelihood function is shown in Eq. 33 and Eq. 34.

$$T(l) = \frac{1}{L} \sum_{k=1}^{k=L} log(\Lambda_k(l)) \gtrless_{H_0}^{H_1} \lambda \tag{33}$$

$$\Lambda_k(l) = P_{YY,k} \tag{34}$$

$$T(l)_{H_0} \approx \sigma_N^2 \tag{35}$$

$$T(l)_{H_1} \approx \sigma_N^2 + SNR \tag{36}$$

The threshold for the likelihood function could be a constant which is higher than the noise PSD. When noise is dynamic however, this can not be a constant. A dynamic value can be chosen based on the previous noise estimation and SNR estimation. A value between σ_N^2 and $\sigma_N^2 + SNR$ can be chosen.

The results of the VAD can be seen in Fig. 9. These results were made using the Noise and SNR estimates from the previous subsystems.

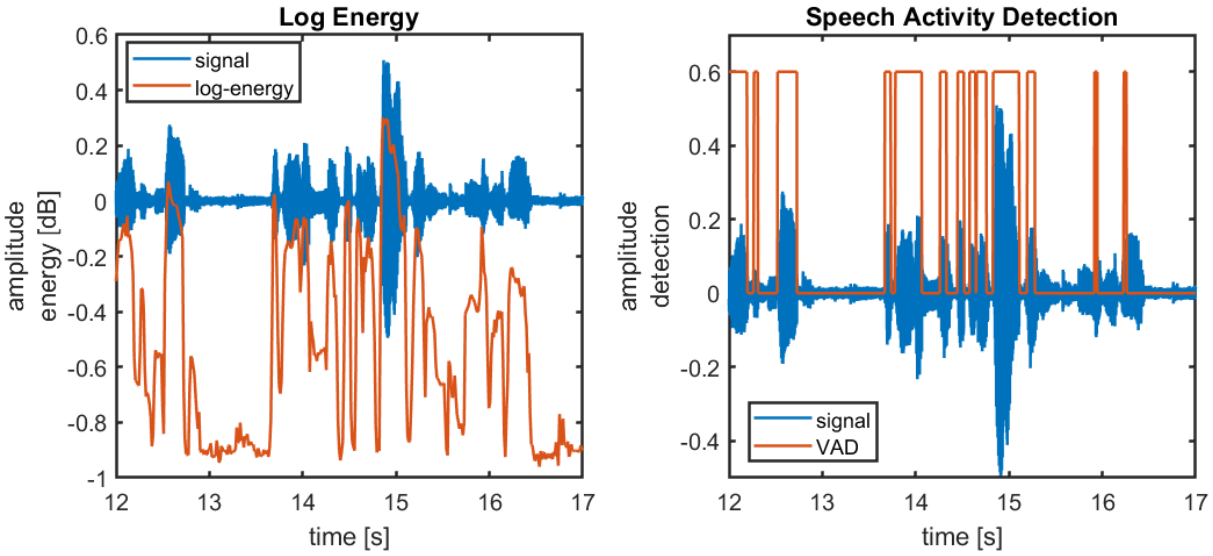

Fig. 9: Log-energy and VAD performance.

VII Target Estimation

Using the information gathered from the different models described in the sections before, the target estimation is determined. This target estimate is then used to go towards then influences the Filter/Gain block. Two different models are discussed and implemented. The first model that will be discussed is the Power Spectral Subtraction.

A. Power Spectral Subtraction

By transforming Eq. 37 into Eq. 38 and doing some manipulations Eq. 39 can be determined. This signal however has no phase associated with it. As the phase has not been changed, the original phase can be multiplied with it to end up with Eq. 40.

$$P_{SS,k}(l) = P_{YY,k}(l) - P_{NN,k}(l) \tag{37}$$

$$\widehat{|S_k(l)|}^2 = |Y_k(l)|^2 - |N_k(l)|^2 \tag{38}$$

$$\widehat{|S_k(l)|} = \left(\max \left\{ 1 - \frac{E\left[|N_k(l)|^2\right]}{|y_k(l)|^2}, \epsilon \right\} \right)^{\frac{1}{2}} |y_k(l)| \tag{39}$$

$$S_k(l) = \widehat{|S_k(l)|} \cdot e^{j \angle y_k(l)} \tag{40}$$

The results with Power Spectral Subtraction implemented can be seen in Fig. 10. Spikes can be seen in the error plot when high the speech signal has high signal power.

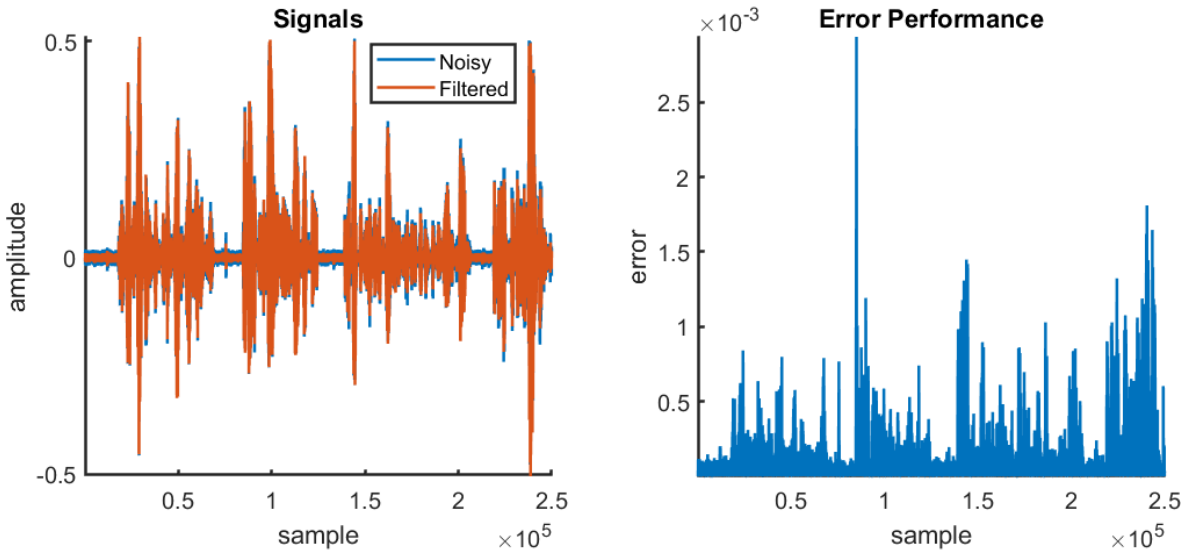

Fig. 10: Results when performing Power Spectral Subtraction using Minimum Statistics Method and SNR_{dd}.

B. *Wiener filter*

The second implemented method is the Wiener filter. The Wiener filter uses the Signal-to-Noise Ratio to determine a gain, Eq. 42 and Eq. 43, with which the noise speech signal is multiplied as can be seen Eq. 41.

$$\hat{S}_k = H_k \cdot Y_k \tag{41}$$

$$H_k = \frac{P_{SY,k}}{P_{PYY,k}} \tag{42}$$

$$= \frac{SNR_k}{SNR_k + 1} \tag{43}$$

The results for the implementation of the Wiener Filter can be found in Fig. 11. The Wiener filter greatly reduces the noise in the parts where no speech is present. However, when speech is present the error seen is much bigger (10 times) than for the Power Spectral Subtraction. Which degrades the Speech Signal.

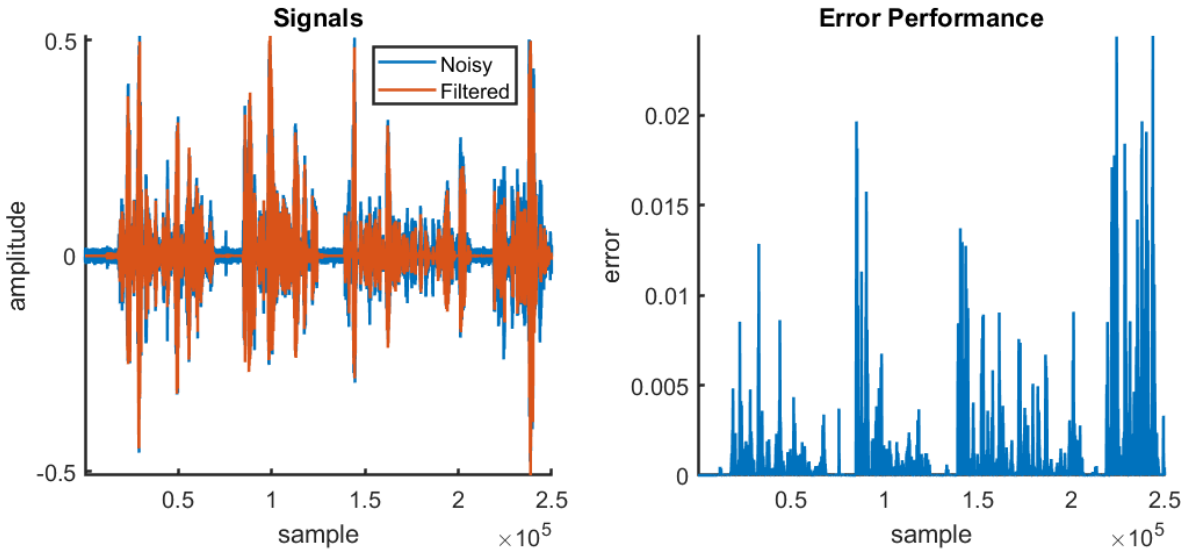

Fig. 11: Results when using the Wiener Filter with Minimum Statistics Method and SNR_{dd}.

VIII Multi-microphone beamforming

When using multiple microphones, the spatial properties of the incomming signals can be exploited. Since there is a (small) but relevant distance difference between the microphones, the time of arrival of the same signal differs. This time delay can be used to estimate the direction of the source and to filter other (interfering) directions.

Before the system is designed and the signals are discussed, some assumptions are made. First of all, the speech and noise are assumed to be WSS. Secondly a far field is assumed where the angles of arrival at every microphone is identical. These two assumptions can be used to simplify the incoming signals at each microphones. Since the signal is WSS, a time delay can be interpreted as a phase shift in frequency.

Because there is a phase shift, spatial aliasing becomes important. This is where distance between microphones become too big where the periodic signal shifts a full time interval. To avoid this, the distance between microphones should be smaller than half of the wavelength. When assuming a maximum frequency of 8000KHz, the maximum distance between microphones is 2 milimiters.

Since the signal only differs in phase and a linear microphone setup, the signal model for each microphone can be defined as in Eq. 44.

$$\mathbf{Y}_k(l) = \begin{bmatrix} S_k(l) & S_k(l)e^{-j2\pi\frac{k\tau}{N}} & \dots & S_k(l)e^{-j2\pi(M-1)\frac{k\tau}{N}} \end{bmatrix} + \mathbf{N}_k(l) \tag{44}$$

In this equation, $\mathbf{Y}_k(l)$ can be expressed as a linear combination of $S_k(l)$ by a steering vector defined in Eq. 45. The resulting model can then be expressed as showed in Eq. 46.

$$\mathbf{d}_k = \begin{bmatrix} 1 & e^{-j2\pi\frac{k\tau}{N}} & \dots & e^{-j2\pi(M-1)\frac{k\tau}{N}} \end{bmatrix} \tag{45}$$

$$\mathbf{Y}_k(l) = S_k(l)\mathbf{d}_k + \mathbf{N}_k(l) \tag{46}$$

From this last model, the source signal can easily opbtained by the equation shown in Eq. 47. This method is called the delay and sum beamformer, since the first signals are delayed untill all the signals can be added. Results and performance of this beamforer is showed in Fig. 12. Where nine microphones were used with a initial SNR of 5 dB and an angle of arival at 90 degrees.

$$\hat{S}_k(l) = \frac{1}{M}\mathbf{d}_k^H\mathbf{Y}_k(l) \tag{47}$$

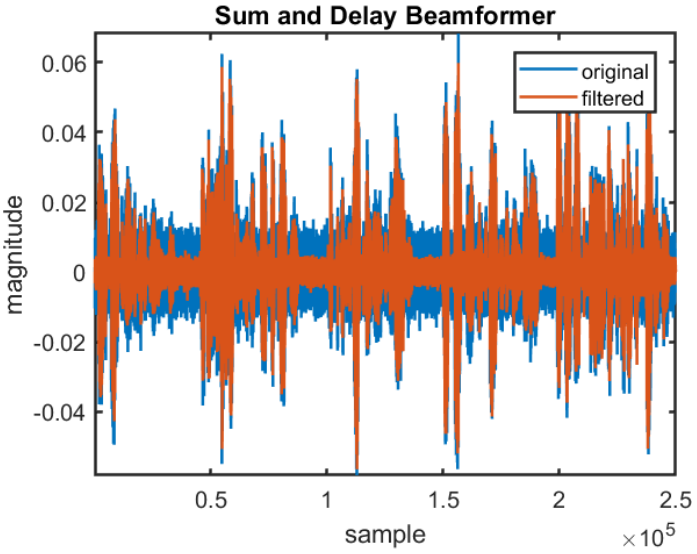

Fig. 12: Performance of the delay and sum beamformer.

IX. Evaluation and Conclusion

A. Evaluation

In this Section, more results and evaluation are discussed about the subsystems and the system as a whole.

The most important part of the estimations, is the noise estimation. In Section IV, three methods of noise estimations are discussed where two are implemented. The MMSE estimator was found to need too many prior knowledge. The other methods, the VAD approach and the MS approach have found to be sufficient to reduce the noise in the system. From observations, the VAD approach seemed more unstable. This was due to the debugging and non-optimal fitting of the variables for the VAD and the SNR estimator on which the VAD depended on. The MS approach was found to be more stable and accurate at estimating the noise. This was due to the smoothing and independence.

The SNR estimator was important for the VAD and the Wiener filter. Two methods were implemented. The maximum likelihood approach and the decicion directed approach. From observations, the smoothening approach (DD), was found to be more effective at estimating the SNR.

At last, the target estimation is discussed. In Section VII, there were two estimators implemented. The Spectral Substraction Method and the Wiener Filter. The Spectral Substraction gave the best absolute error performance and the intelligability was good. However, some noise remained after filtering. The Wiener filter had a better noise reduction, but succeeded less in keeping the original speech signal. This was found in the absolute error evaluation.

A good evaluation of sources is the Mean Opinion Score, where the result is rated 0 to 5. After rating the result, the mean is taken. This is a preference based evaluation and can vary depending on the evaluating party. For this system, the Spectral Substraction received a MOS value of 1.8 while the Wiener filter received a MOS value of 1.2. This shows that noise reduction is not more important than keeping the voice signal clean.

As a bonus, a beamformer was implemented in Section VIII to decrease the static noise on the microphones. This was found to be succesfull depending on the number of microphones. Where more microphones meant a better filtering of the static noise.

B. Conclusion

From the previous evaluation the conclusion was made that the following subsystems would be used for the single channel noise reduction. For the noise estimation, the Minimum Statistics approach was chosen due to its simplicity and performance when K is large. For the target estimation, the spectral substraction method was chosen due to the MOS rating it received. This means that a VAD and SNR estimation would no longer be needed in the system. From this, we can conclude that the system has a low complexity suited for real time audio processing.

References

[1] R. C. Hendriks, T. Gerkmann, and J. Jensen, *DFT-Domain Based Single-Microphone Noise Reduction for Speech Enhancement:A Survey of the State of the Art*. Morgan & Claypool, 2013. [Online]. Available: https://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6813348

APPENDIX A
MATLAB CODE

A. Main

```
clear
%clc
%close all

%% read first file
[x,Fs] = audioread('src/clean_speech.wav'); % audio
x_length = length(x);                       % audio length
frame_length = fix(Fs * 0.018);             % frame length (seconds)
overlap_length = fix(frame_length * 0.6);   % overlap length (percentage)
SNR = 20;                                   % SNR for noise
filter = 'shann';                           % filter

%% Model and segment
y = awgn(x, SNR, 'measured');   % noisy signal
Y = segment(y, frame_length, overlap_length, filter); %Segmentate

%% Memory
xsize = length(Y(:,1));
ysize = length(Y(1,:));

Pnn = zeros(xsize,ysize); %Noise Estimate Power Matrix
Pyy = zeros(xsize,ysize); %Noise signal Power Matrix
S_hat = zeros(xsize,1); %Output
Q = zeros(xsize,ysize); %Q-Matrix for Min Stat
SNRi =  zeros(xsize,1); %SNR of current sample
Speech =  zeros(2,ysize); %SpeechFlags (VAD) stationary and adaptive
SNR = zeros(xsize,ysize); %SNR Matrix
X = zeros(xsize,ysize); %Output X

%% Variables
alpha_noisePSD = 0.98;  % alpha for noise estimate
alpha_SNR = 0.98;       % alpha for SNR estimation
L = 1;                  % dimension of bartlett estimate
QL = 50;                % Amount of samples used for Min of Min Stat vector
QK = 50;              % Amount of samples used for Mean of Min Stat vector
K = 1;                  % asumed stationary for K frames (SNR_ml)
silent_frames = 50;     % number of init silence frames
thresSpec = sqrt(0.1);  % Threshold for Spectral Subtraction
VarNoise = 1;           % 0 is VAD method, 1 is MS method
VarSNR = 0;             % 0 is SNRdd, 1 is SNRml
VarTarget = 1;          % 0 is SSub, 1 is Wiener

tic
%% Noise Enhancement
w = waitbar(0);
for i = 1 : size(Y,2)   % for each frame
    waitbar(i/size(Y,2), w);
    Yi = fft(Y(:,i));   %current frame

    Pyyi = abs(Yi).^2;  %Power of noisy signal frame
    Pyy(:,i) = Pyyi;    %Add to noisy signal power matrix

    if L > 1 && i > L   % bartlett estimate
        for j = 1:frame_length
        Pyyi(j) = sum(Pyy(j,i-L+1:i))./L;
        end
    end

    % Noise PSD Estimation
    if i < silent_frames                %Within the Silent Frames set right flags
        Pnni = abs(Yi).^2;
            if i > 1 && VarNoise == 1   %Fix for first sample MS Method
               Q(:,i) = alpha_noisePSD*Q(:,i-1)+(1-alpha_noisePSD)*Pnni;
            elseif VarNoise == 1        %If MS Method
               Q(:,i) = (1-alpha_noisePSD)*Pnni;
            else                        %If VAD Method
                speechFlag = 0;
                speechFlag2 = 0;
            end
```

```
    else
        if VarNoise == 0                %If VAD Method
            [speechFlag, speechFlag2] = vad(Yi, Pnn(:, i-1), SNRi);
            Pnni = noisePSD(Yi, Pnn(:, i-1), speechFlag2, alpha_noisePSD);
            Speech(1,i) = speechFlag;   %Add VAD stationary to matrix
            Speech(2,i) = speechFlag2;  %Add VAD addaptive to matrix
        else                            %If MS Method
            [Pnni, Q] = MinStat(Pyyi, Q, i, QL, QK, alpha_noisePSD);
        end
    end
    Pnn(:,i) = Pnni;

    % Target PSD Estimation
    if VarSNR == 0
        SNRi = snr_dd(Pyyi, Pnni, S_hat ,alpha_SNR);    %directed decision
    else
        SNRi = snr_ml(Pyyi, Pnni, K);                   %Minimum Likelihood
    end
    SNR(:,i) = SNRi; %Add to SNR Matrix

    % Target Estimation
    if VarTarget == 0   %Spectral Subtraction
        S_hat = spectral_substraction(Pyyi, Pnni, Yi, thresSpec);
    else                %Wiener filter
        S_hat = wiener(Yi, SNRi);
    end

    %Add to X Matrix
    X(:,i) = ifft(S_hat);
end
toc
close(w);
%% Overlap add and sound
xh = overlap_add(X, overlap_length, filter); %Overlap Add back to normal signal
xh = real(xh);                  %Take the real part
xh = xh(1:size(y(:,1),1));  %Remove the extra padding that was added
factor = max(y)/max(xh);    %Matching factor
xh = factor*xh;             %Make xh same height as y

%% plotting
till = 250000; %Plot length
figure

subplot(121); hold on
plot(y(1:till))
plot(xh(1:till))

subplot(122); hold on
e = abs(x-xh).^2;
Pmean = movmean(mean(Pnn).^2, frame_length);
e = e * (max(Pmean)/max(e));
semilogy(e(1:till));
semilogy(linspace(1,till, length(Pnn)),Pmean)
```

B. Functions

1) Segment

```
function Y = segment(y, frame_size, overlap_size, filter)
    if nargin < 4
        disp(1)
       filter = 'hann';
    end

    % calculate the step sizes and number of frames
    step_size = fix(frame_size - overlap_size/2);
    num_frames = fix(length(y) / step_size) + 1;

    pad = num_frames*frame_size - length(y);
    y = padarray(y, pad, 'post');

    % Choose filter
    switch(filter)
        case 'rect'
            h = rectwin(frame_size);
        case 'hann'
            h = hann(frame_size);
        case 'hamm'
            h = hamming(frame_size);
        case 'shann'
            h = sqrt(hann(frame_size));
        case 'black'
            h = blackmanharris(frame_size);
        case 'nutt'
            h = nuttallwin(frame_size);
    end

    % generate the frames
    for i = 0 : num_frames
        index = i*step_size + 1;
        Y(:, i + 1) = y(index: index + frame_size - 1);
    end

    % apply window
    Y = repmat(h, 1, size(Y,2)) .* Y;
end
```

2) vad

```
function [speechFlag, speechFlag2] = vad(Yframe, Pnn_old, SNR)
frame_length = length(Yframe);      % frame length
Pnn_old = log(Pnn_old);
thres = sum(Pnn_old)/frame_length; %Stationary Threshold
%thres = log(sum((abs(Pnn_old).^2)/frame_length));         % threshold= speech log energy
SNRavg = mean(SNR);
thres2 = thres+log(SNRavg)/6; %Adaptive Threshold

E = sum(abs(Yframe).^2) / frame_length;  % Energy
Elog = log(E);                      % Log Energy

speechFlag = Elog > thres;          % one if speech is present
speechFlag2 = Elog > thres2;
end
```

3) noisePSD

```
function noise_estimate = noisePSD(Y, noise_prev, VAD, alpha)
% noise_estimate = frame_sizex1
% Y = frame_sizex1
% noise_prev = frame_sizex1
% VAD = 0 or 1
% alpha = between 0 and 1

    if VAD == 0
        noise_estimate =  noise_prev;
    else
```

```
        noise_estimate =  alpha*noise_prev + (1-alpha)*abs(Y).^2;
    end
end
```

4) MinStat

```
function [Pnn, Q] = MinStat(Pyy, Q, i, L, K, alpha)
    Pnn = zeros(288,1); %Allocate Pnn for output

    Q_new = alpha*Q(i-1) + (1-alpha)*Pyy; %Determine the Q of current Sample
    Q(:,i) = Q_new;     %Add to Q-Matrix
    lbound = i-L+1;     %Determine Lowerbound of Q values used
    if lbound < 1 %Make sure that it is not too low
        lbound = 1;
    end

    for k =1:size(Q,1) %Run for all k frequency bins
        pos = find(Q(k,lbound:i) == min(Q(k,lbound:i))); %Find position of minimum
        mlow = pos(1)+lbound-1-K;   %Mean Low Bound
        if mlow > 1     %If Low Bound is too small
            B = 1/mean(Q(k,mlow:i));
        else
            B = 1/mean(Q(k,1:i));
        end
        Pnn(k,1) = Q(k,pos(1)+lbound-1)*B; %Add to Pnn for kth frequency bin
    end
end
```

5) SNR_ml

```
function [SNR , Pss] = snr_ml(Pyy,Pnn, K)
% Pnn is estimated for each frame
% K is number of frames signal power is equal in previous frames

SNR = Pyy./Pnn - 1; % SNR per frame per frequency
Pss(1:K, :) = Pyy(1:K, :) - Pnn(1:K, :);

    for i = K : size(Pyy, 2)
        Pss(i, :) = sum(Pyy(i-K+1 : i, :) - Pnn(i-K+1 : i, :))/K;
    end

end
```

6) SNR_dd

```
function snr = snr_dd(Pyy, Pnn, S_prev, alpha)
if nargin < 3
   alpha = 0.98;
end

tresh = zeros(length(Pyy),size(Pyy,2));
Pss = abs(S_prev).^2;

    Pssi1 = alpha .* Pss./Pnn;

    Pssi2 = (1-alpha) .* max((Pyy - Pnn),tresh);

    Pss(:,1) = Pssi1 + Pssi2;
    snr = Pss./Pnn;
end
```

7) spectral_substraction

```
function Sk_hat = spectral_substraction(Pyy, Pnn, Yk, thres)
    % allow less arguments
    if nargin < 4
       thres = 0;
    end

    Yk_phase = angle(Yk);
```

```
    Sk_hat = 1-Pnn./Pyy;            % spactral substraction
    Sk_hat(Sk_hat<thres) = thres;   % negative PDF estimates
    Sk_hat = sqrt(Sk_hat).*abs(Yk);
    Sk_hat = Sk_hat.*exp(1i*Yk_phase);
end
```

8) wiener

```
function Sk_hat = wiener(Yi, SNR)


H = SNR./(SNR+1);
Sk_hat = H.*abs(Yi);
Sk_hat = Sk_hat.*exp(1i*angle(Yi));
end
```

9) overlap_add

```
function y = overlap_add(Y, overlap_size, filter)
    % calculate the step sizes and number of frames
    frame_size = size(Y,1);
    step_size = fix(frame_size - overlap_size/2);

    % Choose filter
    switch(filter)
        case 'rect'
            h = rectwin(frame_size);
        case 'hann'
            h = hann(frame_size);
            h(1) = 10^-9;
            h(frame_size) = 10^-9;
        case 'hamm'
            h = hamming(frame_size);
        case 'shann'
            h = sqrt(hann(frame_size));
            h(1) = 10^-9;
            h(frame_size) = 10^-9;
        case 'black'
            h = blackmanharris(frame_size);
        case 'nutt'
            h = nuttallwin(frame_size);
    end
    h = repmat(h, 1, size(Y,2));


    % retrieve signal
    y = zeros(step_size * size(Y,2)-(frame_size-overlap_size), 1);
    h_out = zeros(step_size * size(Y,2)-(frame_size-overlap_size), 1);
    for i = 0 : size(Y,2) - 2
        index1 = i*step_size + 1;
        index2 = index1 + frame_size - 1;
        y(index1 : index2) = y(index1 : index2) + Y(:, i + 1);
        h_out(index1 : index2) = h_out(index1 : index2) + h(:, i + 1);
    end
    h_out = 1./h_out;
    y = h_out .* y;
end
```