Faculty of Power and Aeronautical Engineering

Department of Automatic control & Robotics

Signal processing Project

Noise estimation and subtraction

Prepared by:

Jafar

Pratama Aji Nur Rochman

Noise estimation and subtraction

Task Description

An audio signal is given that contains 2 types of noises: a constant factory noise and an occasional “useful” signal. Roughly determine signal frames without the useful signal and measure the noise in it. Perform noise estimation in the frequency domain and subtract the noise energy from the signal energy. Make visualization of the time-domain signal and its spectrogram, of intermediate and final processing results.

Consider the problem where we want to minimize the noise in a signal, and we can sample the noise component separately. One received channel will comprise the signal plus noise, the other will consist predominantly of noise only. Of course, the noise component is not precisely, sample - by - sample, identical to that received by the signal sensor. If it were, the problem would reduce to a simple sample - by - sample subtraction [1].

The spectral subtraction method is a simple and effective method of noise reduction. In this method, an average signal spectrum and an average noise spectrum are estimated in parts of the recording and subtracted from each other, so that the average signal-to-noise ratio (SNR) is improved [2].

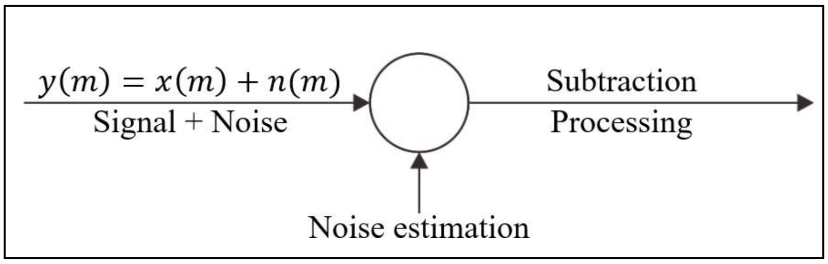

Figure 1. Illustration

The situation is illustrated in Figure 1. The noise source is correlated to some degree with the measured signal and represent this coupling via a linear transfer function. If possible, to estimate the linear transfer function, the noise that can estimate as received by the signal sensor, and thus subtract it out.

Algorithm Description

The noisy signal y(m) is a sum of the desired signal x(m) and the noise n(m):

$$y(m) = x(m) + n(m) \qquad (1)$$

In the frequency domain, this may be denoted as:

$$Y(f) = X(f) + N(f) \qquad (2)$$

where Y(f), X(f) and N(f) are the Fourier transforms of the noisy signal y(m), x(m) and n(m) the original signal x(m) and the noise n(m) respectively, and f is the frequency variable.

In spectral subtraction, the incoming signal x(m) is buffered and divided into segments of N samples length. Each segment is windowed, using a Hamming window, and then transformed via discrete Fourier transform (DFT) to N spectral samples. The windowed signal is given by:

$$y_w(m) = w(m)y(m) = w(m)[x(m) + n(m)] = x_w(m) + n_w(m) \qquad (3)$$

And this operation will be represented in frequency domain as:

$$Y_w(f) = W(f) * Y(f) = X_w(f) + N_w(f) \quad (4)$$

where the operator * denotes convolution.

The equation describing the spectral subtraction is expressed as:

$$|\hat{X}(f)|^2 = |Y(f)|^2 - \alpha\overline{|N(f)|^2} \qquad (5)$$

where $|\hat{X}(f)|^2$ is an estimate of the original signal spectrum $|X(f)|^2$ and $\alpha\overline{|N(f)|^2}$is the time-averaged noise spectral. α controls the amount of noise subtracted from the noisy signal. For full noise subtraction, α=1 and for over-subtraction α>1

The time-averaged noise spectrum is obtained from the periods when the signal is absent and only the noise is present as:

$$\overline{|N(f)|^2} = \frac{1}{k}\sum_{i=0}^{k-1}\overline{|Ni(f)|^2} \qquad (6)$$

The statistic parameters of the noise are not known; thus the noise and the speech signal are replaced by their estimates:

$$\hat{X}(f) = \hat{Y}(f) - \hat{N}(f) \qquad (7)$$

The noise spectrum estimate $\hat{N}(f)$ is related to the expected noise spectrum E[$|N(f)|$] which is usually calculated using the time-averaged noise spectrum $\overline{N}(f)$ taken from parts of the recording where only noise is present. The noise estimate is given by:

$$\hat{N}(f) = |\tilde{N}(f)| = \lambda_n \cdot |\tilde{N}_{k-1}(f)| + (1 - \lambda_n) \cdot |\tilde{N}_k(f)| \qquad (8)$$

where $\hat{N}(f)$ is the smoothed noise estimate in i-th frame, λ_n is the filtering coefficient ($0.5 \leq \lambda_n \leq 0.9$, some authors use values $0.8 \leq \lambda_n \leq 0.95$). To obtain the noise estimate, the part of the recording containing only noise that precedes the part containing speech signal should be analyzed (the length of the analyzed fragment should be at least 300 ms). To achieve this, additional speech detector has to be used.

The spectral subtraction error may be defined as:

$$\varepsilon \overset{\text{def}}{=} \hat{X}(f) - \mathrm{X(f)} \qquad (9)$$

This error degrades the signal quality, introducing the distortion known as residual noise or musical noise. The error is a function of expected $E[|N(f)|]$ or average $\bar{N}(f)$ noise spectrum estimate:

$$\varepsilon = N(f) - E[|N(f)|] = \bar{N}(f) - E[|N(f)|] \qquad (10)$$

Therefore, the longer noise section is used in analysis, the more accurate the noise estimate is.

The signal-to-noise ratio may be defined in frequency domain as SNR a priori (for clean signal) or SNR a posteriori (for noisy signal). SNR in k-th frame is given by:

$$SNR_k^{\alpha\, prior} \stackrel{\text{def}}{\equiv} \frac{|X_k(f)|^2}{|N_k(f)|^2} \qquad (11)$$

$$SNR_k^{\alpha\, posterior} \stackrel{\text{def}}{\equiv} \frac{|Y_k(f)|^2}{|N_k(f)|^2} \qquad (12)$$

During the restoration process, the clean signal is not known, hence the SNR a priori value has to be estimated. Using the Gaussian model, optimal SNR in k-th frame may be defined as:

$$\overline{SNR}_k^{\alpha\, prior} = (1-\eta)\,.P(SNR_k^{\alpha\, posterior} - 1) + \eta\,.\frac{|\hat{X}_{k-1}(f)|^2}{var(N_k(f))^2} \qquad (13)$$

$$SNR_k^{\alpha\, posterior} \stackrel{\text{def}}{\equiv} \frac{|Y_k(f)|^2}{var(N_k(f))^2} \qquad (14)$$

Where P(x):

$$P(x) = \begin{cases} x\,;\ for\ x \geq 0 \\ 0\,;\ otherwise \end{cases}$$

$var(N_{k-1}(f))$ is the variance of the noise spectrum in the previous frame, $\hat{X}(f)$ is estimate of the restored signal and η is constant ($0.9 < \eta < 0.98$). The variance is usually replaced by spectral power of noise estimate.

Solution

1. Load Audio

The first thing to do is load the audio file to be processed, the goal at this stage is to read the audio that will be processed, and the audio will be processed into a signal so that the duration (time) and frequency of the signal will also be obtained.

```
input_path  = 'glas-11025-fixed.wav';

function [y, fs, dt] = load_audio(input_path)
    [y, fs] = audioread(input_path);
    N = length(y);
    t = N / fs;
    dt = linspace(0, t, N);
end
```

After the audio file is successfully loaded, then the signal wave that still contains noise is displayed. By adding some parameters, alpha is used for full noise subtraction, beta, and gamma for magnitude spectral subtraction 2 for power spectrum subtraction

```
[ym, fs, t] = load_audio(input_path);
subplot(2,1,1);
plot(t, ym);
title('Useful signal + noise y(m)'); ylabel('Amplitude'), xlabel('time [s]');

[yn, fn, tn] = load_audio(noise_path);
subplot(2,1,2);
plot(tn, yn);
title('noise n(m)'); ylabel('Amplitude'), xlabel('time [s]');
```

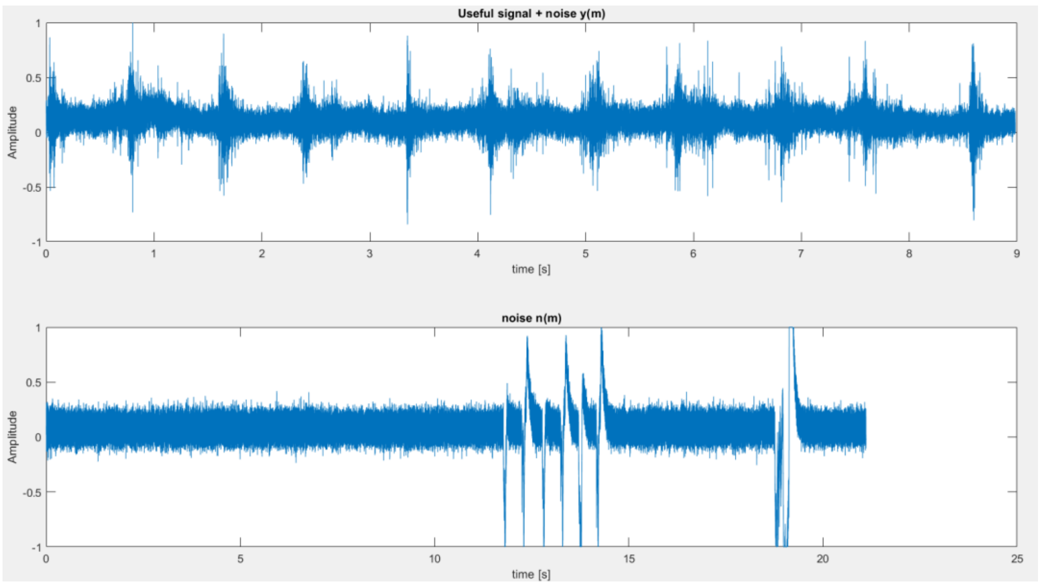

Figure 3. Useful signal and noise

2. Segmentation Signal

he purpose of segmentation is to calculate how many segments there will be, this number is the longitude of the signal minus the number of windows that divides the per cent shift and creates a W*N matrix i.e. the matrix containing our segmentation with N signals of length Based on 4th formula that $Y_w(f)$ (windowed function in domain time is equal to $W(f) * Y(f)$ (window in domain time convoluted with the signal y(m) in time domain). So, a function was created to given y(m)(signal), fs (frequency signal), wl (Window Length) and SP (Overlap percent) will be obtain $Y_w(f)$ and W. In this function there are several tasks that will be done such as calculating the number of windows this is equal to the length of each window (W) multiplied per the frequency of the signal, obtaining y_w(m) by using the segment function and applying Fast Fourier Transform

```
function [YfW,W] = segmentation(ym,fs,wl,SP)
    W = fix(wl*fs);
    Hw = hamming(W);
    L = length(ym);
    SP = fix(W.*SP);
    N = fix((L-W)/SP +1);
    Index=(repmat(1:W,N,1)+repmat((0:(N-1))'*SP,1,W))';
    hw=repmat(Hw,1,N);
    y=ym(Index).*hw;
    YfW=fft(y,W);
End
```

3. Spectrogram

Based on 5th formula, $|\hat{X}(f)|^{\gamma} = |Y(f)|^{\gamma} - \alpha\overline{|N(f)|^{\gamma}}$ where $|\hat{X}(f)|^{\gamma}$ is an estimate of the original signal spectrum $|X(f)|^{\gamma}$,$\alpha\overline{|N(f)|^{\gamma}}$is the time-averaged noise spectra and $|Y(f)|^{\gamma}$ is the average signal spectrum of the noisy signal, then in order to estimate the original signal spectrum needed to first calculate the average signal spectrum and for this needed to do the spectrogram of Y(f). To compute the average signal spectrum, where basically segment by segment compute the average of the previous, the current and the next frame.

```
function YfAverage = AverageSignalSpectrum(YfSpectogram)
    numberOfFrames=size(YfSpectogram,2);
    for i=2:(numberOfFrames-1)
        YfAverage(:,i)=(YfSpectogram(:,i-1)+YfSpectogram(:,i))/2;
    end
end

[yn, fn, tn] = load_audio(noise_path);
[YfW_n, W_n] = segmentation(yn, fn, wind_length, overlap);
YfSpectogram_n = abs(YfW_n(1:fix(end/2)+1,:)).^Gamma;
YfAverage_n = AverageSignalSpectrum(YfSpectogram_n);

[YfW, W] = segmentation(ym, fs, wind_length, overlap);
YfSpectogram = abs(YfW(1:fix(end/2)+1,:)).^Gamma;
YfAverage = AverageSignalSpectrum(YfSpectogram);

[YfW_n, W_n] = segmentation(yn, fn, wind_length, overlap);
YfSpectogram_n = abs(YfW_n(1:fix(end/2)+1,:)).^Gamma;
YfAverage_n = AverageSignalSpectrum(YfSpectogram_n);
```

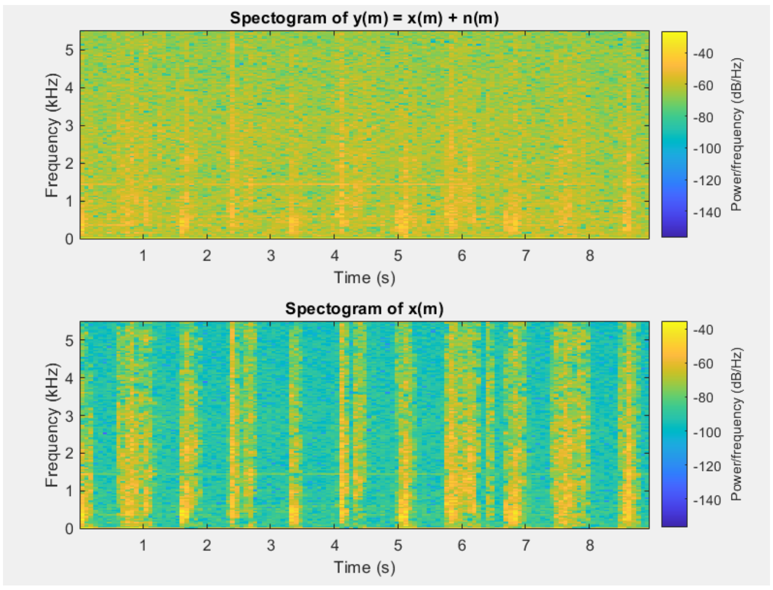

Figure 4. Spectrogram

4. Spectral Subtraction

A function was created given the Spectrogram (containing $X_w(f) + N_w(f)$, from here will be extract the time-averaged noise spectra($\overline{|N(f)|^\gamma}$) , the Average Signal Spectrum ($|Y(f)|^\gamma$) , frequency , number of windows, Alpha (For full noise subtraction, α=1 and for over-subtraction α>1.) and Beta (To attenuate the noise). For the first frame to the last one realizing the following process, to call all the Noise Estimator function. If the frame only contains noise this will be storage to do the noise estimation will be performed and updated, and our $X_w(f)$ value will be equal to $\beta|Y(f)|$ that means that we will multiply our noisy frame for a Beta that will attenuate the noise in this frame.

```
function X =
    SpectralSubtraction(YfSpectogram,YfAverage,fs,W,Alpha,Beta,Gamma,YfAverage_n)
    SP=.4;
    IS = .25;
    NIS=fix((IS*fs-W)/(SP*W) +1);
    numberOfFrames=size(YfSpectogram,2);
    N=mean(YfAverage_n(:,1:NIS)')';
    NCounter=0;
    NLength=9;

    for i=1:numberOfFrames-2
       [SFlag, NCounter, ~]=NoiseEstimator(YfSpectogram (:,i).^(1/Gamma),
    N.^(1/Gamma), NCounter);
       if SFlag==0
             N=(NLength*N+YfSpectogram(:,i))/(NLength+1);
             X(:,i) = Beta*YfSpectogram(:,i);
       else
             D=YfAverage(:,i)-(Alpha*N);
             X(:,i)=max(D,0);
       end
    end
```

```
end
```

The Signal-to-Noise Ratio is performed if the value of SNR is smaller than a given threshold then we know that the frame contains just noise.

```
function [ SFlag, NCounter, Dist]=NoiseEstimator(signal,noise,NCounter)
    NMargin = 3;
    Hangover = 8;
    SpectralDist= 20*(log10(signal)-log10(noise));
    SpectralDist(SpectralDist<0)=0;

    Dist=mean(SpectralDist);

    if (Dist < NMargin)
       NCounter=NCounter+1;
    else
       NCounter=0;
    End

    if (NCounter > Hangover)
       SFlag=0;
    else
       SFlag=1;
    end
end

Xf = SpectralSubtraction(YfSpectogram, YfAverage, fs, W, Alpha, Beta, Gamma,
    YfAverage_n);
```

5. Reconstruction Signal

Function was created to reconstruct the signal given the function $|\hat{X}(f)|^{\gamma}$, the previously calculated Phase of Y(f) and the Window length.

```
function ReconstructedSignal=Reconstruct(x,yphase,wl)
    SP = .5;
    SLength = SP * wl;

    [FreqResolution, FrameNumber]=size(x);
    yphase = yphase(:,1:897);
    csv=x.*exp(1i*yphase);

    if mod(wl,2)
        csv=[csv;flipud(conj(csv(2:end,:)))];
        csv=[csv;flipud(conj(csv(2:end-1,:)))];
    end
    rsig=zeros((FrameNumber-1)*SLength+wl,1);
    for i=1:FrameNumber
        startposition=(i-1)*SLength+1;
        spectrum=csv(:,i);
        rsig(startposition:startposition+wl-1)=rsig(startposition:startposition+wl-
    1)+real(ifft(spectrum,wl));
    end
    ReconstructedSignal=rsig;
End

figure();
plot(t,ym); hold on; plot(t(1:98780),xm); title('Original vs Specter-abstracted
    signal');xlabel('Time [s]');
ylabel('Amplitude'); xlim([0 max(t)]);
```

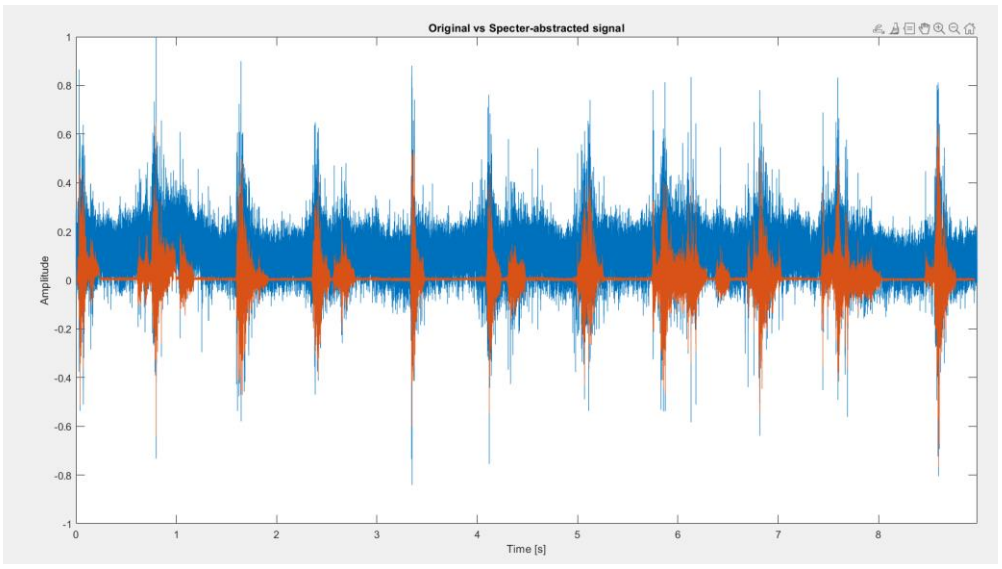

Figure 5. Signal with noise and reconstructed signal without noise

<u>References</u>

1. LEIS, J. W. (2011). Digital Signal Processing Using MATLAB for Students and Researchers. John Wiley & Sons.

2. Steven W. Smith. The Scientist and Engineer's Guide to Digital Signal Processing. Second Edition, California Technical Publishing, San Diego, CA, 1999

3. <u>https://sound.eti.pg.gda.pl/denoise/noise.html</u>