Noise Reduction in Speech

Siyan Wan Yuqing Li
4793757 4835417
Statistical Digital Signal Processing and Modeling
EE4C03 Assignment
14-11-2018

***Abstract*—In the acoustic world, noise is almost ubiquitous. The speech signals can easily be affected by noise from diverse sources. Infections like those will change the characteristics of the speech signals and decrease the speech quality and intelligibility, which cause considerable damage to human to human or human-to-machine communication systems. Therefore, speech enhancement of the noisy speech signal is required.**

Noise reduction in speech usually can be formulated mathematically. The noisy speech signal passes a linear filter, then a clean speech estimation can be obtained. In this project, we design an optimal filter, Wiener filter, to reduce noise in speech without noticeable speech distortion.

***Keywords*— Wiener filter, noise reduction**

I. Introduction

The speech signal in real life is inevitably infected by the surrounding environment, the noise introduced by the media, the noise brought by the mechanical transmission, the electrical noise generated by the communication device itself, and other noises. These background noises will seriously affect the quality and intelligibility of the original speech signal. When the speech signal is disturbed or even flooded by various noises, speech enhancement is needed to extract useful speech signals from the noise background and to suppress and reduce noise interference. In a word, it extracts the clean original speech from the noisy speech. For the listener, it is mainly to improve the voice quality, improve the voice intelligibility and reduce the fatigue; for the speech processing system (identifier, vocoder, mobile phone), it is to improve the recognition rate and anti-interference ability of the system. It is related to the theory of speech signal processing, and it involves human auditory perception and phonetics.

There are many sources of noise, and they have different characteristics depending on the application. Therefore, it is difficult to find a general-purpose speech enhancement algorithm that can be applied to various noise environments. Different speech enhancement strategies must be adopted for noise in different environments. One of the effective ways to achieve this goal is to design a filter with the best linear filtering characteristics. When the noise containing speech signal passes through the filter, it can reproduce the signal as accurately as possible or make the most accurate estimate possible. The Wiener filter is one of the typical representatives of such a filter. Wiener filter is a filter to produce an estimate of a desired random process by linear time-invariant filtering of a measurement noise. The Wiener filter can minimize the mean square error between the estimated random process and the desired process.

II. Wiener Filter Modeling

In order to recover a signal $d(n)$ from the noisy measurement. We can think of the speech signal as a set of digital signals and the Wiener filter as a linear system. The illustration of Wiener filter is shown in Fig. 1.

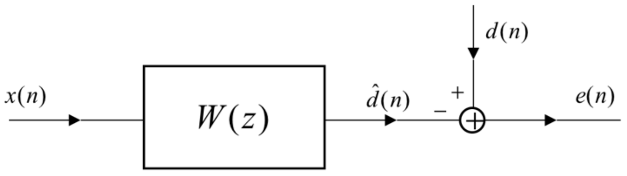

Fig. 1. Illustration of Wiener filter

$x(n)$ is the unprocessed speech signal that contains noise. It is assumed that both $v(n)$ and $d(n)$ are wide-sense stationary and they are uncorrelated.

$$x(n) = d(n) + v(n) \tag{2.1}$$

Since the goal is to establish a filter so that when $x(n)$ passes through the linear filter $w(n)$ and the result $y(n)$ is as close as possible to $d(n)$. So, $y(n)$ is called the estimated value of $d(n)$, i.e.

$$y(n) = \hat{d}(n) \tag{2.2}$$

In fact, the convolutional form shown in equation (2.2) can be understood as the current and past observations $x(n), x(n-1) \dots x(n-p+1)$ to estimate the current value of the signal $d(n)$. Therefore, using $w(n)$, the filtering problem is actually a statistical estimation problem.

$$\hat{d}(n) = \sum_{l=0}^{p-1} w(l)x(k-l) \tag{2.3}$$

Therefore, Wiener filters are often known as optimal linear filtering and prediction or linear optimal estimation. The so-called optimal is based on the minimum mean square error.

If we denote the true and estimated values of the signal by $d(n)$ and $\hat{d}(n)$ respectively, we use $e(n)$ to represent the error between them, i.e.

$$e(n) = d(n) - \hat{d}(n) \tag{2.4}$$

Apparently, $e(n)$ may be positive or negative and it is a random variable. Therefore, it is reasonable to use its mean square error to express the error. The so-called mean square error is the smallest statistical expectation of its square:

$$\xi = E\{|e(n)|^2\} \tag{2.5}$$

The reason why the minimum mean square error criterion is used is that its theoretical analysis is relatively simple and does not require a description of the probability.

In order to determine the filter coefficients, $w(k)$, according to the minimum mean square error criterion shown in equation (2.5), derivative of ξ to $w^*(k)$ should be equal to zero.

$$\frac{\partial \xi}{\partial w^*(k)} = 0 \tag{2.6}$$

With

$$e(n) = d(n) - \sum_{l=0}^{p-1} w(l)x(n-l) \tag{2.7}$$

It follows that

$$\frac{\partial e^*(k)}{\partial w^*(k)} = -x^*(n-k) \tag{2.8}$$

Now we get Wiener-Hopf equations

$$\sum_{l=0}^{p-1} w(l) r_x(k-l) = r_{dx}(k) \tag{2.9}$$

$$\mathrm{R}_X w = r_{dx} \tag{2.10}$$

It is assumed that the $v(n)$ is zero mean noise and it is uncorrelated with $d(n)$. Therefore,

$$E\{d(n)v^*(n-k)\} = 0 \tag{2.11}$$

The cross-correlation between $d(n)$ and $x(n)$ becomes

$$r_{dx}(k) = r_d(k) \tag{2.12}$$

For the same reason, the autocorrelation of $x(n)$ follows that

$$r_x(k) = r_d(k) = r_v(k) \tag{2.13}$$

Fourier transform of the equation (2.12) and (2.13), we get

$$P_{dx}(\omega) = P_x(\omega) \tag{2.14}$$

And

$$P_x(\omega) = P_d(\omega) + P_v(\omega) \tag{2.15}$$

So, the Wiener filter frequency response is

$$W(\omega) = \frac{P_{dx}(\omega)}{P_x(\omega)} = \frac{P_d(\omega)}{P_d(\omega)+P_v(\omega)} \tag{2.16}$$

Above is the model of Wiener filter, we can compute the filter coefficients by estimating one part of the dataset and then use the other part to verify the filter.

III. EXPERIMENT PROCESS

In this project, we use the Wiener filter method to reduce the noise of the speech signal. We use MATLAB to achieve our algorithm of speech enhancement (See appendix for code).

We use the small subset of TIMIT dataset to design (train) and test our speech enhance algorithm. 20 speech segments are used as training data to get the weights of Wiener filter. Other 65 speech segments are used as testing data to use the Wiener filter to estimate the clean signal.

Our speech enhancement algorithm can be divided into two parts: the training part and the testing part. For the training part, we take the speech signal from training dataset as the clean signal and add random noise into the clean signal to generate the noisy signal (unprocessed speech signal that contains noise). After calculating the autocorrelation matrix of noisy signal and the cross-correlation vector, we can get the Wiener filter according to the Wiener-Hopf equations (1.10). For the testing part, we also use the clean speech signal from testing dataset to generate the noisy signal with the settable SNR. The parameters of Wiener filter designed in the last part can be used to estimate the clean speech signal. We compare the SNR of the processed signal with the SNR of the noisy signal to evaluate the performance of our speech enhancement algorithm.

IV. CONCLUSION

In this project, we consider noise reduction in speech as a digital filtering problem, where the clean speech estimation is obtained by passing the noisy speech through a linear filter. With such a formulation, we design an optimal filter in MATLAB to enhance the speech. From the result, we can see that after filtering, noise can be significantly suppressed without noticeable speech distortion.

The performance of our speech enhancement algorithm can be evaluated by SNR. In this project, we set the SNR of the noisy signal as 5dB, while the SNR of the processed signal is about 7.1 dB in the testing part. Comparing these two signals noise ratio, it is observed that the noise can be reduced after filtering.

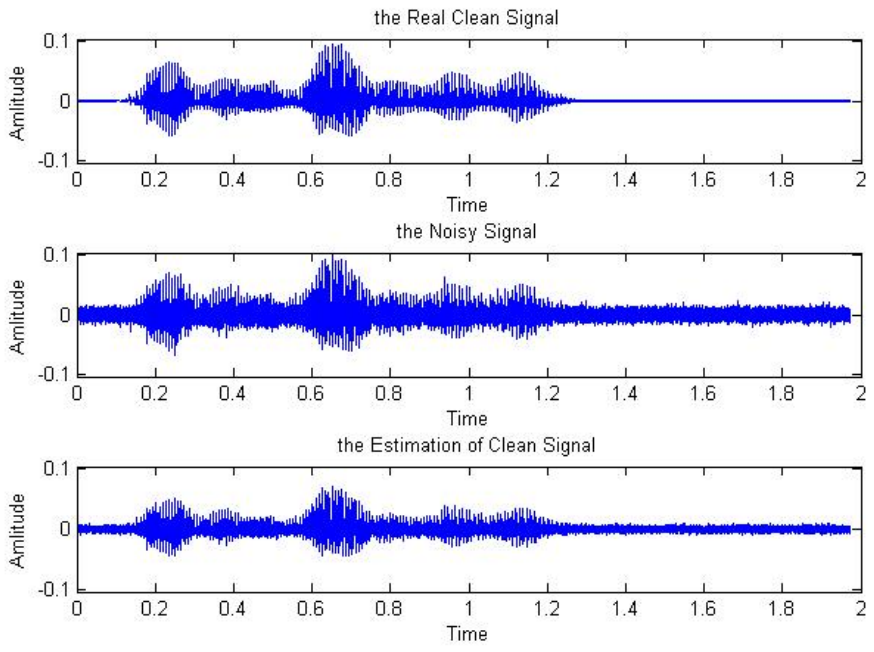

Fig. 2. Time-domain waveform of the clean signal,
the noisy signal and the estimation of the clean signal

To make the result more intuitive, we compared the oscillograms of the clean signal, the noisy signal and the estimation of the clean signal (Fig. 2). Moreover, we also compared the autocorrelations and the periodograms of these three signals (Fig. 3, Fig. 4 and Fig. 5). It is observed that the processed speech signal can estimate the clean speech signal well.

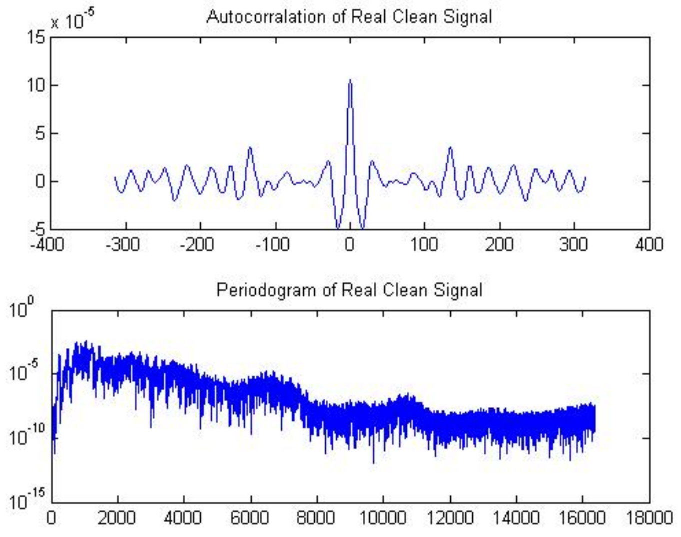

Fig. 3. The autocorrelation and the periodogram of the clean speech signal

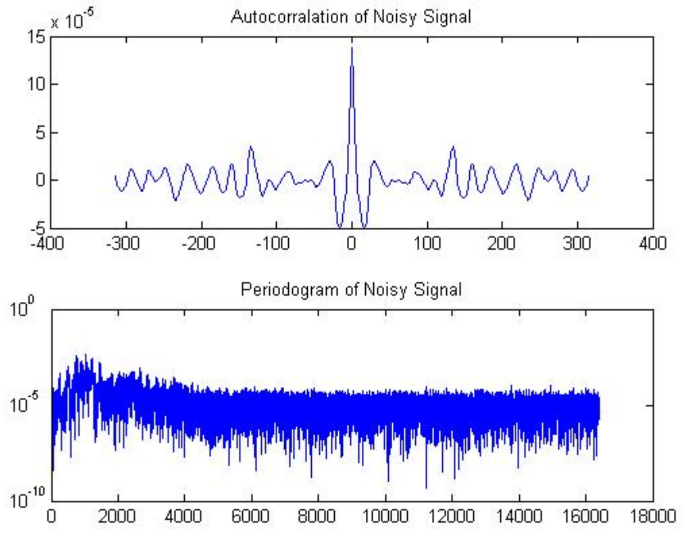

Fig. 4. The autocorrelations and the periodograms of the noisy signal

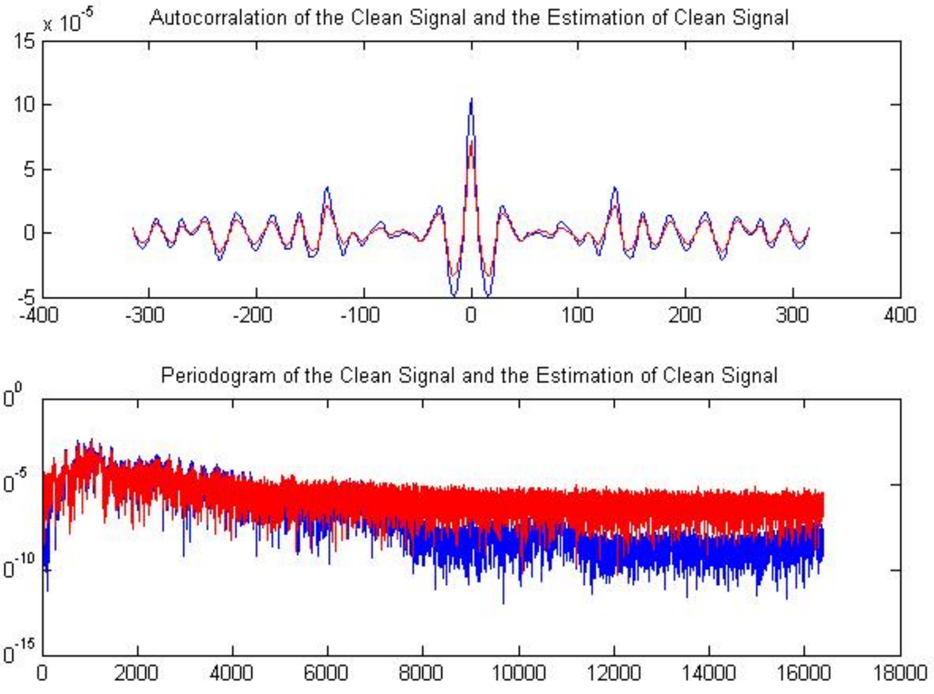

Fig. 5. The autocorrelations and the periodograms of the clean signal
and the estimation of the clean signal
(The blue lines are the clean signal and the red line are the estimation of the clean signal)

From the result, we can see that after filtering, noise can be significantly suppress without noticeable speech distortion.

REFERENCE

[1] Gruber M J . Statistical Digital Signal Processing and Modeling[J]. Technometrics, 1997, 39(3):2.

APPENDIX

1. SpeechEnhancement_train.m

```
%% Speech Enhancement train
%**************************************************
% Training dataset: *.wav
% We use the signals in training dataset as clean singles
% By adding noise in clean signals, we can generate the noisy singals.Changing the variable SNR can
change the SNR of noisy signal.
%

PathRoot='C:\Users\E\Desktop\DSP-program-
SpeechEnhancement\SpeechEnhancement\demo4\wavtrain\'; % Pathroot of training dataset
list=dir(fullfile(PathRoot));
fileNum=size(list,1)-2;
wiener_filter_h=zeros(fileNum,200);
SNRs=zeros(1,fileNum);

for k=3:fileNum
    i=k-2;
    filename =strcat(PathRoot,'/',list(k).name);
    [Input, Fs] = audioread(filename);
    Time = (0:1/Fs:(length(Input)-1)/Fs)';
    Input = Input(:,1);                                  % Choose single channel

    SNR=5;                                               % SNR is the ratio of clean signal and noise
    [NoisyInput,Noise] = add_noise(Input,SNR);       % Generating the noisy signal. NoisyInput is noisy
signal, and Noise is noise.

    %% Compute wiener filter for each training signal
    [wiener_enspeech,h] = wiener(NoisyInput,Input);
    h=h';
    wiener_filter_h(i,:)=h;



    % Normalization of the signal length
    sig_len=length(wiener_enspeech);
    NoisyInput=NoisyInput(1:sig_len);
    Input=Input(1:sig_len);
    wiener_enspeech=wiener_enspeech(1:sig_len);
    Time = (0:1/Fs:(sig_len-1)/Fs)';

    %% Compute the SNR of each training siganl filted by wiener filter
    SNR_Wiener=snr(Input,Input-wiener_enspeech);
    SNRs(i)=SNR_Wiener;
```

```
end

SNR_train=mean(SNRs);                          % Average SNR of filted training dataset
wiener_filter=mean(wiener_filter_h,1);         % Choose the average of h(n) as estimation of wiener
filter h(n)
csvwrite('Filter.csv',wiener_filter)           % save the estimation of wiener filter h(n) in .csv file
```

2. add_noise.m

```
function [Y,NOISE] = add_noise(X,SNR)
% X is clean signal，SNR is required SNR of noisy signal (dB)，Y is noisy signal
% NOISE is the noise added into clean signal
NOISE=randn(size(X));
NOISE=NOISE-mean(NOISE);
signal_power = 1/length(X)*sum(X.*X);
noise_variance = signal_power / ( 10^(SNR/10) );
NOISE=sqrt(noise_variance)/std(NOISE)*NOISE;
Y=X+NOISE;
end
```

3. wiener.m

```
function [wiener_enspeech,h] = wiener(NoisyInput,Input)
xn = NoisyInput';
x = Input';

Mlag = floor((length(NoisyInput)-1)/100);
Rxn=xcorr(xn,Mlag,'biased');                   % Autocorrelation function of noisy
signal xn

%% Wiener Filter
N=200;   %the length of wiener
n0 = Mlag+1;
Rxnx=xcorr(xn,x,Mlag,'biased');                %Cross-correlation function of noisy
signal xn and clean signal x

rxnx=zeros(N,1);
rxnx(:)=Rxnx(n0:n0+N-1);
Rxnxn=zeros(N,N);                              %Generate the autocorrelation
matrix of noisy signal xn
Rxnxn=diag(Rxn(n0)*ones(1,N));
for i=2:N
    c=Rxn(n0+i)*ones(1,N+1-i);
    Rxnxn=Rxnxn+diag(c,i-1)+diag(c,-i+1);
end
```

```
h=zeros(N,1);
h=inv(Rxnxn)*rxnx;                                          %compute the wiener filter h(n)
yn=filter(h,1,xn);                                          %filtering noisy signal xn

wiener_enspeech = yn';
end
```

4. SpeechEnhancement_test.m

```
%% Speech Enhancement test
%**************************************************
% Testing dataset: *.wav
% We use the signals in testing dataset as clean singles
% By adding noise in clean signals, we can generate the noisy singals. Changing the variable SNR can
change the SNR of noisy signal.
%
PathRoot='C:\Users\E\Desktop\DSP-program-
SpeechEnhancement\SpeechEnhancement\demo4\wavDataset\';    % Pathroot of testing dataset
list=dir(fullfile(PathRoot));
fileNum=size(list,1)-2;
h=csvread('Filter.csv');                        % Read the h(n) of wiener filter


%% test multiple signals in testing dataset
for k=3:fileNum
    i=k-2;
    filename =strcat(PathRoot,'/',list(k).name);
    [Input, Fs] = audioread(filename);
    Time = (0:1/Fs:(length(Input)-1)/Fs)';
    Input = Input(:,1);                          % Choose single channel

    SNR=5;                                       % SNR is the ratio of clean signal and noise
    [NoisyInput,Noise] = add_noise(Input,SNR); % Generating the noisy signal. NoisyInput is noisy
signal, and Noise is noise.


    %% Wiener Filtering
    wiener_enspeech = filter(h,1,NoisyInput);


    % Normalization of the signal length
    sig_len=length(wiener_enspeech);
    NoisyInput=NoisyInput(1:sig_len);
    Input=Input(1:sig_len);
    wiener_enspeech=wiener_enspeech(1:sig_len);
    Time = (0:1/Fs:(sig_len-1)/Fs)';

    wiener_filter_h(i,:)=h;

    %% Compute the SNR of each testing siganl filted by wiener filter
    SNR_Wiener=snr(Input,Input-wiener_enspeech);
    SNRs(i)=SNR_Wiener;
end
SNR_test=mean(SNRs);
SNR_test;

%% test single signal
filename =strcat(PathRoot,'/',list(fileNum).name);
[Input, Fs] = audioread(filename);
%% Autocorralation and Periodogram of clean signal
M = floor((length(Input)-1)/100);
Rs = xcorr(Input,M,'biased');
Pss = periodogram(Input);

figure(1)
subplot(2,1,1)
plot((-M:M),Rs)
title('Autocorralation of Real Clean Signal')

subplot(2,1,2)
semilogy(Pss)
title('Periodogram of Real Clean Signal')

%% Autocorralation and Periodogram of noisy signal
M = floor((length(NoisyInput)-1)/100);
Ry = xcorr(NoisyInput,M,'biased');
Pyy = periodogram(NoisyInput);

figure(2)
subplot(2,1,1)
plot((-M:M),Ry)
title('Autocorralation of Noisy Signal')

subplot(2,1,2)
semilogy(Pyy)
title('Periodogram of Noisy Signal')
% Normalization of the signal length
sig_len=length(wiener_enspeech);
NoisyInput=NoisyInput(1:sig_len);
```

```
Input=Input(1:sig_len);
wiener_enspeech=wiener_enspeech(1:sig_len);
Time = (0:1/Fs:(sig_len-1)/Fs)';

%% Compare the clean signal, noisy signal and filtered signal
figure(3)
MAX_Am(1)=max(Input);
MAX_Am(2)=max(NoisyInput);
MAX_Am(3)=max(wiener_enspeech);
subplot(3,1,1);
plot(Time, Input)
ylim([-max(MAX_Am),max(MAX_Am)]);
xlabel('Time')
ylabel('Amlitude')
title('the Real Clean Signal')

subplot(3,1,2);
plot(Time, NoisyInput)
ylim([-max(MAX_Am),max(MAX_Am)]);
xlabel('Time')
ylabel('Amlitude')
title('the Noisy Signal')

subplot(3,1,3);
plot(Time, wiener_enspeech)
ylim([-max(MAX_Am),max(MAX_Am)]);
xlabel('Time')
ylabel('Amlitude')
title('the Estimation of Clean Signal')

%% Autocorralation and Periodogram of filtered signal
M = floor((length(Input)-1)/100);
Rs = xcorr(Input,M,'biased');
Pss = periodogram(Input);

Rso = xcorr(wiener_enspeech,M,'biased');
Psso = periodogram(wiener_enspeech);

figure(4)
subplot(2,1,1)
plot((-M:M),Rs,'b')
title('Autocorralation of the Clean Signal and the Estimation of Clean Signal')
hold on
plot((-M:M),Rso,'r')
```

```
hold off

subplot(2,1,2)
semilogy(Pss,'b')
title('Periodogram of the Clean Signal and the Estimation of Clean Signal')
hold on
semilogy(Psso,'r')
hold off
```