

EE432: Digital Signal Processing Fall 2011

Project 4: Voice PreProcessing

Assigned: Tues 9/13/2011

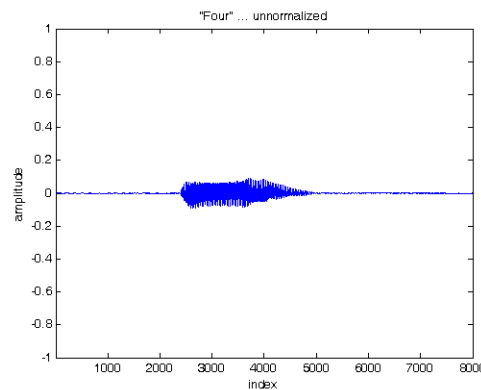
Due: Wed 9/21/2011

Introduction

In this lab, you will write some functions that will be used to preprocess real data signals, such as voice signals. You will use the voice .wav files you collected in the last project, and to begin, we will assume that each .wav file contains one spoken word.

I. MATLAB Functions

1. Typically in a voice recording, there are gaps in between words and sounds. When processing voice for word or speaker recognition, it is not always useful to process the “dead” time. An example is shown in the figure below:

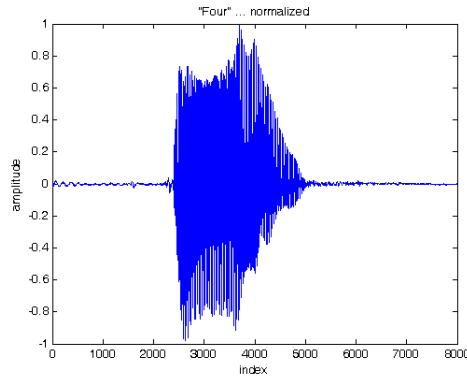


Write a MATLAB function called *FindSignalStart* that will determine when the speech signal actually begins in an input speech vector (there may be some dead time before a voice is heard, see the figure on the next page):

```
>> [y, index]=FindSignalStart(x)
```

where x is the input vector of samples. The output y is the speech sample that is the same input but with the first non-speaking samples (background noise) removed; that is, the spoken word begins right at the beginning of y . The output $index$ is the index number in the input vector where you determined the word actually begins.

Depending on the microphone and other factors, recordings can be done at low signal levels (voltage). When you normalize the signal, there's a better chance to correctly determine the starting point. Below you see the same signal from the previous plot after it has been normalized to fall in the range $[-1,+1]$.



For this project, the signal should be normalized using your *normalize432* function, then squared to make all values positive before finding the starting index (this effectively converts your signal into a power signal). You can use a threshold value on the power signal to determine where the signal starts. The *find* command may prove useful here.

Important considerations:

How do you tell if a word is present in a speech signal? What threshold would you apply to the power signal? Hint: Plot some of the word files to get an idea.

- Write a MATLAB function called *FindSignalEnd* that will determine when the signal ends in an input speech vector (there may be some dead time after the word is spoken:

```
>> [y, index]=FindSignalEnd(x)
```

where x is the input vector of speech samples. The output y is the speech sample that is input but with the last non-speaking samples (background noise) removed; that is, the spoken word begins right at the beginning of y . The output $index$ is the index number in the input vector where you determine the word actually ends. You should use the same ideas you had for the *FindSignalStart* function. The *flipud* or *fliplr* function may prove useful here.

Important considerations:

How you tell when a word ends is similar to finding when a word begins.

- Write a MATLAB function called *AddNoise* that will add Additive White Gaussian Noise (AWGN) to an input signal:

```
>> y=AddNoise(x, sigma)
```

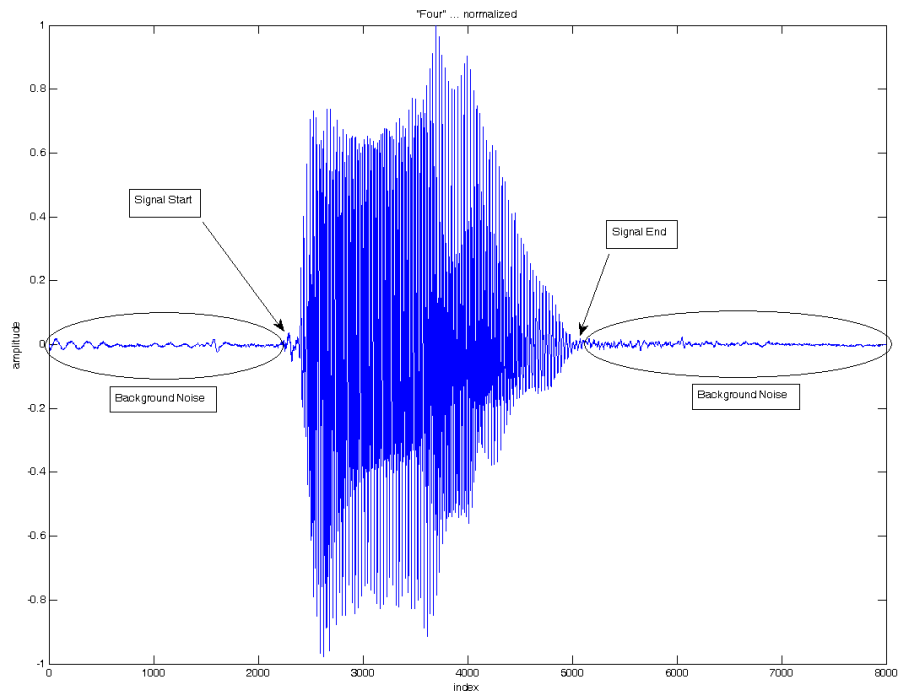
where x is the input signal. Noise should be added using the *randn* function. The output y is the input signal + noise. The second input $sigma$ is the standard deviation of the noise you add. Use zero-mean noise, so that the noise power is equal to $sigma^2$.

- Write a MATLAB function called *ComputeBkgdNoise* that will compute the statistics of the background noise in an input noisy signal. These will be used in S/N calculations. It is used as in:

```
>> [Nmean, Nstd]=ComputeBkgdNoise(x)
```

where x is the input vector. The outputs $Nmean$ and $Nstd$ are the estimated mean value of the noise and its standard deviation. Compute the background noise statistics in a portion or portions of the input vector where there is no word being spoken. One way to do this is to strip off the background noise before the word is spoken, and the background noise after the

word is spoken, and compute the mean and standard deviation of the noise. Note that your *FindSignalStart* and *FindSignalEnd* functions can help you do this.



II. Test Your Functions

1. Choose one of each of your spoken digit .wav files (0-9), use `wavread` to read in the data, and remove the beginning and ending silence. Play each digit with the beginning and ending silence removed to see how well your functions above work. Also, plot the signal and visually compare the index that your *FindSignalStart* and *FindSignalEnd* functions determined to be the start with the beginning that your eye tells you. If they don't match too well, you need to adjust how you are determining the beginning and end of the signal.
2. Repeat the previous step, except create one long speech signal by joining all of the digits (w/silence removed) together. Play this long vector...it should sound like someone saying "zero-one-two-three-four-five-six-seven-eight-nine" without pauses in between each word.

Write out this file using `wavwrite` and email it to the professor. Be sure that after you use `wavwrite`, you use Windows to listen to the file so it is what you expect.

3. For the following files, record the time (in seconds) when each speech signal began and ended USING YOUR *FindSignalStart* AND *FindSignalEnd* FUNCTIONS.

File	Begin Time (sec)	End Time (sec)
raven7c.wav		
Drizzle.wav		

Did your functions work well on these signals?

4. Add zero-mean Gaussian noise to one of your “Should we chase?” speech clips. Recall that the *randn* function can be used to create Gaussian noise. Put in enough noise so that the words can be barely distinguished when you listen to it. Record the following:

Max value of the “Should we chase?” signal: _____

Min value of the “Should we chase?” signal: _____

Your value of noise standard deviation for just-distinguishable words: _____

Write out this noisy file using *wavwrite* and email it to the professor. Be sure that after you use *wavwrite*, you use Windows to listen to the file so it is what you expect.

5. For the following files, record the background noise mean and standard deviation using your *ComputeBkgdNoise* function:

File	Noise Mean	Noise Standard Deviation
raven7c.wav		
Drizzle.wav		

For this lab, turn in your well-commented code, email the two .wav files specified, and answer questions 3 - 5 above.