24.900 Squib: Exploration of Acoustic Phonetics

Tony Kim

December 7, 2007

1 Introduction

This paper describes my experimentation with acoustic phonetics. It consists of two distinct parts. They are:

1. The implementation of my own sound wave analyzer in Matlab, and its calibration against Praat[1], a standard computer software for phoneticians in the analysis of speech.
2. The use of Praat to investigate the distinction between foreign and native pronounciation of certain Korean words. This term, I had served as a source for two 24.900 classmates, and have also conducted fieldwork on the Mongolian language. Hence, I have seen on many occasions where one is unable to reproduce some sound "correctly" (both on my part and on the parts of my interviewers.) I investigate this "failure" phonetically.

In particular, on the latter point, I conclude that typical spectograms are very misleading!. When comparing two spectograms, although we can often find significant variations in the higher frequency ranges, it is always the sub-2000Hz region that contains the acoustic elements that differentiate between speakers.

The above content reflects two different motivations for this study. On one hand, I was interested in applying signal analysis techniques that I had informally encountered in my other studies. In addition, I also had interest in bringing an objective arbitrator to identify the differences in the pronounciation of words produced by two speakers.

2 Implementation of a Sound Wave Analyzer

In Matlab, the speech signal is represented as a vector of length n, where n is the total number of data points that corresponds to the sample. (So, n is also the product of the sampling rate and the total duration.) When plotted in sequence, the signal vector can be visualized as a complicated waveform as in Figure (1).

The objective of my sound wave analyzer, then, is to take such a signal, and to present the variations in frequency content over the duration of the sample. In other words, we wish to produce a spectogram.

[1]The software Praat can be accessed at: http://www.fon.hum.uva.nl/praat/

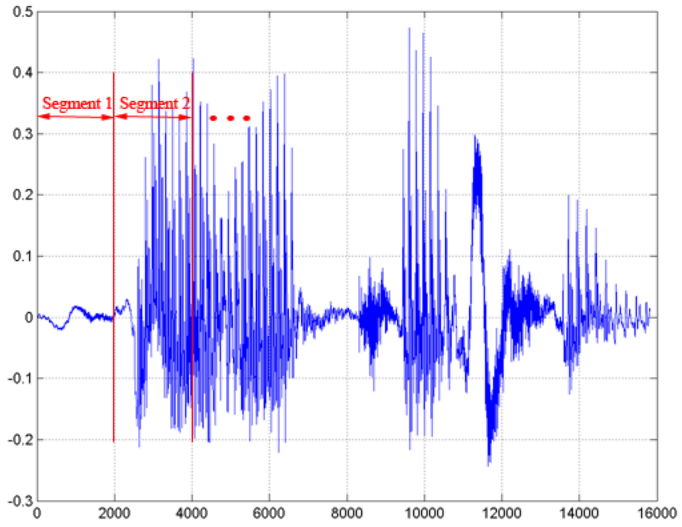


Figure 1: The sound wave corresponding to my pronounciation of the word "phonetician."

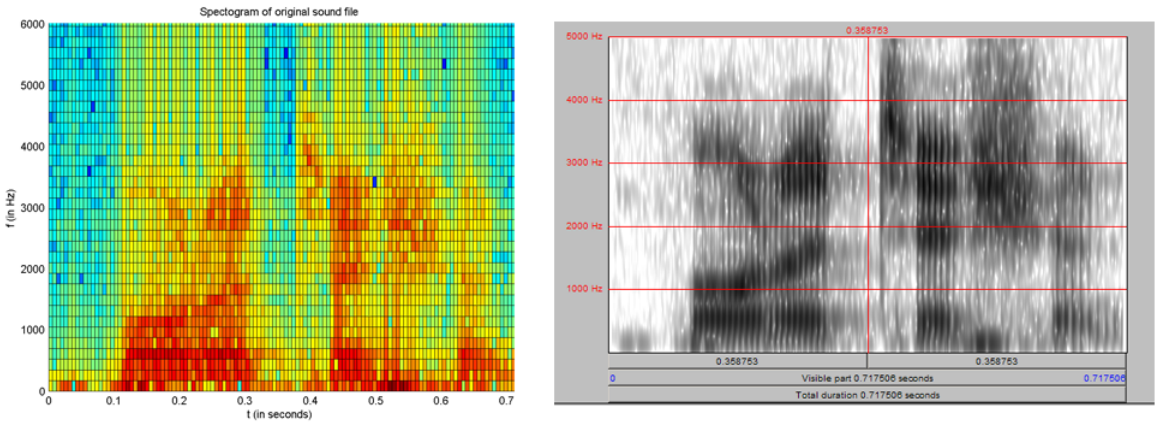
2.1 Spectogram generation

We achieve this task through the Fourier transform. In this discussion, we do not consider the transform in any detail.[2] It is sufficient to note that given some signal as a function of time, the Fourier transform is a mathematical operation that yields the amplitudes (and phase) of the sinusoids with various frequencies that compose the signal. A well-known result in Fourier analysis, called Rayleigh's energy formula[3], then shows that the magnitude squared of the transform represents the energy distribution of the signal over the frequencies present.

Because we are interested in producing the energy distribution as a function of time, the technique is to take a speech signal, such as my pronounciation of the word "phonetician", and to divide it into N equal segments (See Figure (1)). We then take each segment and apply the Fourier transform. By this procedure we obtain N frequency distribution profiles, each of which corresponds to the particular segment. By stringing these over time, we can obtain a spectogram of the signal. Since additional details on this computation would not be appropriate for this paper, we move onto the results. (Rather, in Appendix A, I have attached my Matlab scripts for sound wave analysis. The code should be more-or-less self-explanatory. However, I have also included additional discussion alongside the code, which detail the issues encountered while working on the program.)

Figure (2) is a side-by-side comparison of my spectogram to that generated by Praat, a standard program for acoustic phonetics. There is good correlation between the regions of high intensity in the two results, although it is clear that Praat offers much better resolution. Due to its better performance, for the remainder of the assignment, I will use Praat for the generation of spectograms, while using my Matlab code for resynthesis of sounds.

[2]My reference of choice is "Applications of Discrete and Continuous Fourier Analysis" by H. Joseph Weaver.
[3]See: Pg. 206 in "Dr. Euler's Famous Formula" by P. Nahin (Not as silly a book as the title might suggest!)



(a) (b)

Figure 2: (a) The spectogram of the word "phonetician" as computed by my Matlab script. Red and blue indicate regions of high and low intensity, respectively. (b) The same sound file as analyzed by Praat, where darkness represents intensity.

2.2 Filter and resynthesis

The ability to view the frequency content of a signal allows new kinds of signal manipulation. For instance, we are now able to address questions such as:

- What formants in the frequency spectrum are *required* for the correct perception of speech? (Answer: None in particular. This was the first experiment I attempted.)
- What acoustic signature distinguishes the pronounciations of a word by a native and a foreign speaker?

For the former, we are able to apply a restrictive filter to the sound file, to see if the synthesized output is comprehensible. For instance, I have "chopped up" the spectogram of "phonetician" into 200 Hz windows, beginning at 200 Hz and ending at 5000 Hz.[4] As it turns out, all of these windows (in the lower frequencies, at least) are recognizable as pronounciations of "phonetician." Appendix B gives information on how to access these sound files.

For the latter, we may be able to *delete away* possible differences in two pronounciations, again using filters. If the resulting filtered output "sounds the same" to human listeners, we may be justified in arguing that the omitted acoustic characteristic was responsible for the differentiation of the two audio samples. This is the strategy I follow in the next section, where I attempt to identify acoustic elements responsible for differences in native and non-native pronounciation of Korean words.

[4]In other words, I have applied the following band-pass filters 200-400Hz, 400-600Hz, ..., 4800-5000Hz on the original sound file. Separate file for each, of course.

3 Native and Non-native Pronounciations of Korean

Over the term, I served as a Korean speaker for two of my 24.900 classmates, Mike Vasquez and Kathryn Lin. (Neither are native Korean speakers.) During these sessions I had noticed that there were several words that both had difficulty reproducing. The examples given here are the Korean words "daughter", "rice", and "cat." In this section, I present the spectograms of these words as pronounced by me and Mike. For consistency, my pronounciations always appear on the left, and Mike's on the right.

3.1 "Daughter"

The Korean word for "Daughter" goes as: /θ^{h}al/.

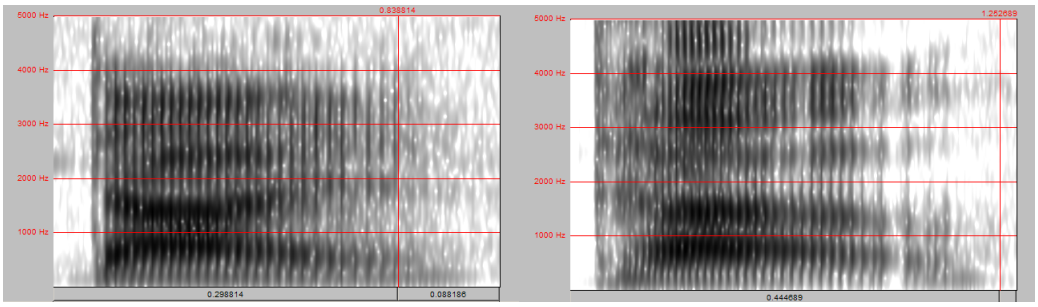


Figure 3: Unprocessed spectogram of /θ^{h}al/. Left is Tony. Right is Mike.

With this example, I wish to illustrate explicitly the reasoning used to identify the "differentiating" region of the spectogram. From Figure (3) it is clear that the following are sites of notable differences:

- In Mike's speech, there is a dark band above 4000 Hz at the beginning of the sample. This is notably absent in my speech.
- In my speech, there is an emerging band approximately at 2500 Hz. This is not so prominent in Mike's.
- There is a major difference in the shape of the bottom two bands. In particular, in my speech, the bands have nonzero initial slopes (positive for the 800 Hz band, and negative for the 1500 Hz band); whereas in Mike's the two are relatively flat.

I wish to argue that the the difference in perception originates from the bottom two bands. As remarked before, we achieve this by selectively deleting certain portions of the sound sample, and comparing the resynthesized outputs. Of course, the arguments of this section is most convincing when *hearing* these results. Appendix B gives the directions to the relevant files.

To demonstrate the above claim, I have produced sound clips from the original files, using the following windows (i.e. band-pass filters):

- 0 - 1000Hz
- 1000 - 2000Hz

- 0 - 2000Hz
- 2000 - 4000Hz

To show the correctness of my synthesis code, I present the Praat analysis of the synthesized output for the 2000 - 4000Hz window in Figure (4). The other synthesized files have been similarly verified for their correctness.

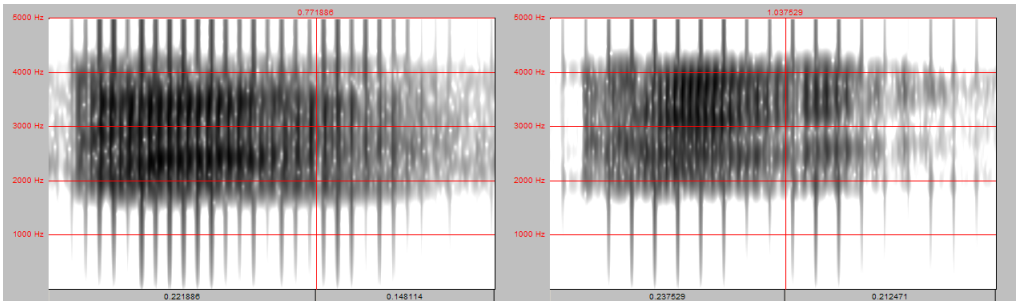


Figure 4: Spectogram of /θ^{h}al/ on the 2000 - 4000Hz window. I'm not sure why the striations are present.

Upon reviewing the output files, it should be clear that the 2000 - 4000Hz results are only negligibly different. On the other hand, the difference in perception is distinct for the first three filters. It then logically follows that *both* bands in the window 0 - 2000 Hz are important for the distinction between Mike's and my pronounciations.

3.2 "Rice"

We begin with the spectograms of /s^{h}al/ ("rice").

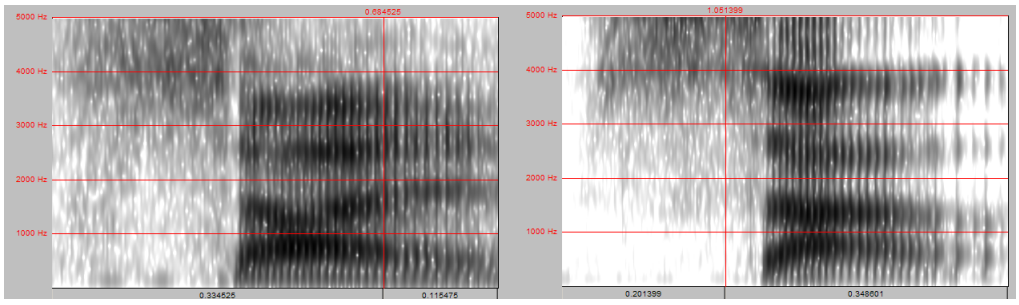


Figure 5: Spectogram of /s^{h}al/.

Interestingly, we note that the lowest formant (800 Hz) in both samples appear to be similar. This is verified in the *sound* of the 0 - 1000Hz window outputs. Using the same frequency domains as in the previous example, we find again that the higher frequency deviations do *not* contribute to the auditory difference.

3.3 “Cat”

Lastly: the Korean pronounciation of “cat” is /gow jaŋ i/.

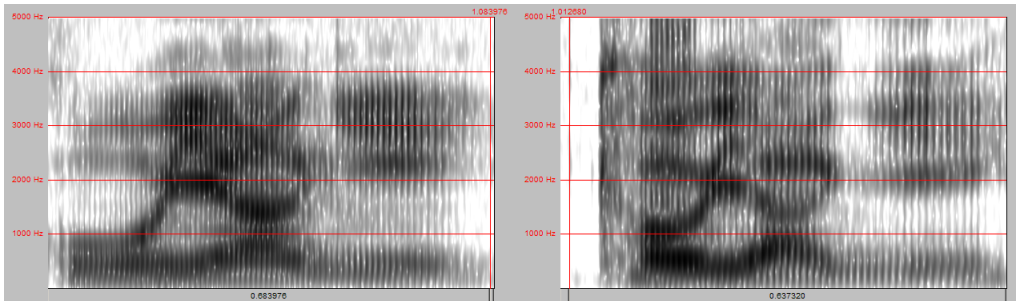


Figure 6: Spectogram of /gow jaŋ i/.

The structure presented here is considerably more complicated than the previous two. Hence, I have attempted many different pass regions. Upon reviewing the results, it should be clear that, again the 0 - 2000 Hz window is most important for the differential perception.

4 Conclusion

In this study, I have first constructed a program that allows for the study of speech signals by producing their spectograms. In addition, this program is able to apply a bandpass filter on the results, and to resynthesize the filtered output.

On the second part, I have used this tool to investigate the phonetic differences between native and non-native pronounciations of Korean words. In the examples given, the spectogram analysis indicates that there are many differing features in the sound signals. However, using our filtering technique, we have shown that the difference in perception always originates from the sub-2000Hz range. Put in another way, we have seen that the first two formants of the spectrum contain all the information regarding the speech content.

A Matlab code for Sound Wave Analyzer

A.0.1 Main body

```
%-------------------------------------------------
% Main code for spectogram generation
% Example usage:
%   [orig synth Fs] = main('phonetician.wav',200,500);
%-------------------------------------------------

function [y z Fs] = main(filename,filterlow,filterhigh)
%-------------------------------------------------
% Load sound file
%-------------------------------------------------
[y Fs nBits] = wavread(filename);
sound(y,Fs);
Y = length(y);

%-------------------------------------------------
% Knobs
%-------------------------------------------------
N_desired = 80;

L = floor(Y/N_desired);
L = 2^(nextpow2(L)-1);

%-------------------------------------------------
% Cut the sound stream into parts each of length L
%-------------------------------------------------
s = split(y,L);
N = length(s(:,1)); % Actual number of partitions

%-------------------------------------------------
% Prepare spectogram
%-------------------------------------------------
f = Fs/2*linspace(0,1,L/2);

S = zeros(N,L);
for i = 1:N
    S(i,:) = fft(s(i,:),L)/L;

    plot(f,log(2*abs(S(i,1:L/2))));
    drawnow;
end

%-------------------------------------------------
% Visualize spectogram
%-------------------------------------------------
```

```
t     = linspace(0,Y/Fs,N);
[X Y] = meshgrid(t,f);
% Decide on linear vs. log plot
%surf(X,Y,2*abs(S(:,1:L/2))');
surf(X,Y,log(2*abs(S(:,1:L/2)))');
xlabel('t (in seconds)');
ylabel('f (in Hz)');
title('Spectogram of original sound file');

%------------------------------------------------
% Apply filter
%------------------------------------------------
F = zeros(1,L);

% Pass region
F = zeros(1,L);
a1 = floor(filterlow/(Fs/L))
a2 = floor(filterhigh/(Fs/L))
if (a1 ~= a2)
    F(max(a1,1):a2) = 1;
end

for i = L/2+1:L-1
    F(i) = F(L-i);
end

for i = 1:N
    S(i,:) = F.*S(i,:);
end

figure;
surf(X,Y,log(2*abs(S(:,1:L/2)))');
xlabel('t (in seconds)');
ylabel('f (in Hz)');
title('After applying filter');

%------------------------------------------------
% Resynthesized
%------------------------------------------------
SS = zeros(N,L);
for i = 1:N
    SS(i,:) = L*ifft(S(i,:),L);
end
z = appendv(SS);

figure
subplot(211);
plot(real(z));
hold on;
```

```
plot(imag(z),'r');

subplot(212);
plot(y);
sound(real(z),Fs);
z = real(z);

% Save the result
filename = filename(1:length(filename)-4);
outFile = strcat(filename,'_',int2str(filterlow),'_',int2str(filterhigh));
wavwrite(z,Fs,outFile);
```

A.0.2 Helper functions

```
%-------------------------------------------------
% Split:
% Takes a vector and fragments it into smaller
% vectors each of length L
%-------------------------------------------------
function s = split(v,L)

M = length(v);
N = floor(M/L);

s = zeros(N,L);
for i = 1:N
    for j = 1:L
        s(i,j) = v(L*(i-1)+j);
    end
end


%-------------------------------------------------
% Appendv:
% Takes a matrix and collapses its elements into
% a row vector
%-------------------------------------------------
function s = appendv(v)

M = length(v(1,:));
N = length(v(:,1));
L = N*M;

s = zeros(1,L);
for i = 1:N
    s((1+M*(i-1)):(M*i)) = v(i,:);
end
```

B How to access the sound data

For the second part of this paper, the argument rests on being able to *hear* the results of our filters on the original sound file. All files relevant to this paper (i.e. Matlab scripts) can be found at:

`http://web.mit.edu/kimt/www/24.900/squib/`

As for the specific speech samples:

- "Phonetician":

  `http://web.mit.edu/kimt/www/24.900/squib/phonetician/`

- Korean words (original recording):

  `http://web.mit.edu/kimt/www/24.900/squib/miketony_originals/`

- "Daughter":

  `http://web.mit.edu/kimt/www/24.900/squib/miketony_daughter/`

- "Rice":

  `http://web.mit.edu/kimt/www/24.900/squib/miketony_rice/`

- "Cat":

  `http://web.mit.edu/kimt/www/24.900/squib/miketony_cat/`

The file naming scheme follows the format:

`filename_filterlow_filterhigh.wav`

where:

- filterlow is the lower limit of the bandpass filter.
- filterhigh is the upper limit of the bandpass filter.

So, for example,

`‘‘phonetician_200_400.wav’’`

is the result of applying on "phonetician.wav" a filter that lets only the frequencies between 200Hz and 400Hz through.