EEC 201: Final Project Report

Abhinav Kamath, Mason del Rosario

March 16, 2019

Contents

1 Background

1.1 Voiced and Unvoiced Sounds

Human speech is comprised of **voiced** and **unvoiced** sounds. Voiced sounds leverage the vocal cords, which oscillate at specific frequencies, while unvoiced sounds mostly rely on quick bursts of air through the mouth. As a result, the voiced portions of speech are characterized by high-amplitude frequency content around specific frequencies (or "pitches") while the unvoiced portions are characterised by low-amplitude white noise. Figure 1 showcases this phenomenon.

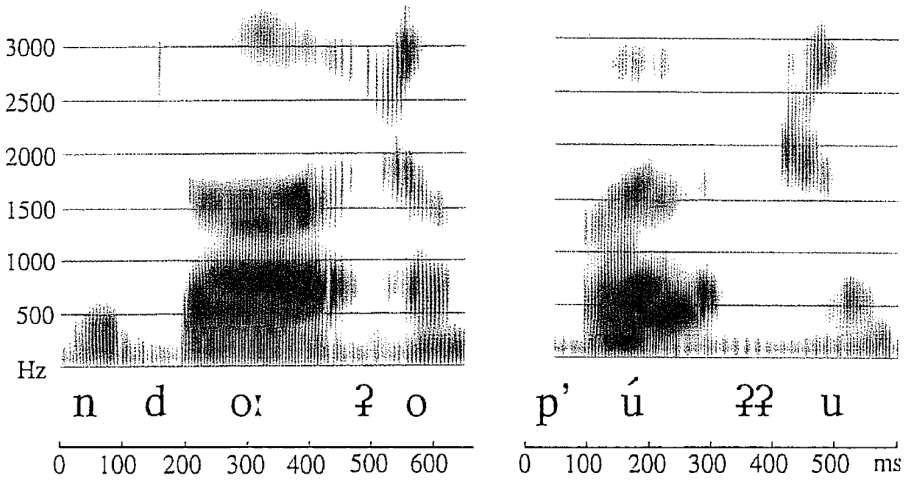

Figure 1: Spectrograms for two words in the Dahalo language [3]. Voiced sounds (e.g., 'o' and 'ù') have higher magnitude at their respective frequencies while unvoiced sounds (e.g., 'n' and 'd') have lower magnitude.

1.2 Linear Predictive Coding

In analyzing a speech sample, the signal is segmented and analyzed piecemeal such that voiced/unvoiced regions can be distinguished from one another. Linear Predictive Coding (LPC) of speech leverages this framework in attempting to synthesize an estimate of the original audio. Assuming that the human vocal system is a linear filter, $A_k(z)$, excited by a train of impulses, $e[n]$, the problem of synthesis can be reduced to determining the coefficients of the filter and the periodicity of the impulse train.

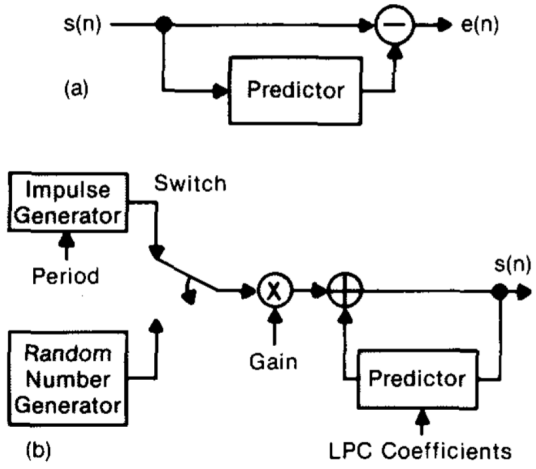

Figure 2: Block diagram showing methodology for LPC [5]. In synthesizing the audio signal, $s[n]$, the objective is two-fold: 1) Find the LPC coefficients which define the 'predictor' filter, 2) construct the residual signal, $e[n]$, which is comprised of periodic impulses (voiced sounds) and random white noise (unvoiced sounds).

2 Methods

2.1 Analysis: Autocorrelation Method

For a given windowed portion of a speech sample, the window can be modeled as an LTI system excited by a periodic impulse train. This can be seen in (1) where $y[n]$ is the *windowed speech sample*, $e[n]$ is the *excitation pulse train*, and $a_k[n]$ is the *synthesis filter*.

$$y[n] = e[n] \circledast a_k[n] \tag{1}$$

$a_k[n]$ is an LTI system whose z-transform can be modeled as an all-pole IIR filter, $A_k(z)$, which is written in (2).

$$A_k(z) = \frac{1}{a_p z^p + a_{p-1} z^{p-1} + \cdots + a_1 z + a_0} \tag{2}$$

The coefficients, a_k, of the all-pole LPC model can be found by using the **autocorrelation method** [1] as shown in (5).

$$\mathbf{r}(j) = \sum_{k=1}^{p} \mathbf{r}(j-k) a_k \tag{3}$$

$$R\mathbf{a} = \mathbf{r} \tag{4}$$

$$\mathbf{a} = R^{-1}\mathbf{r} \tag{5}$$

Where p is the order of the desired LPC model. $\mathbf{r(j)}$ denotes the autocorrelation of the windowed speech sample, $y[n]$. The matrix, R, is a Toeplitz matrix whose entries are comprised of $R(j)$, both of which are shown in (6).

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} R = \begin{bmatrix} R(0) & R(1) & \dots & R(p-1) \\ R(1) & R(0) & \dots & R(p-2) \\ \vdots & & & \vdots \\ R(p-1) & R(p-2) & \dots & R(0) \end{bmatrix}, \mathbf{r} = \begin{bmatrix} R(1) \\ R(2) \\ \vdots \\ R(p) \end{bmatrix} \tag{6}$$

Thus, the LPC characterization of a speech sample can be accomplished by taking the following steps:

1. Segment the speech sample into frames with an appropriate window.
2. On a frame-by-frame basis, perform the autocorrelation on each frame.
3. Construct R and $\mathbf{r}$ to find the vector of LPC coefficients, $\mathbf{a}$.

2.2 Synthesis: Pitch Detection

Given the fame-wise synthesis filter as described above in (2), the main goal in synthesizing an approximation of the original speech sample is to determine the instantaneous pitch of the excitation pulse train, $e[n]$. This builds on our assumption that speech can be modeled as written in (1). In this work, we take two approaches to estimating the pitch of $e[n]$: the cepstrum-based method and the autocorrelation method.

2.2.1 Method #1: Cepstrum-based Pitch Detection

Taking the cepstrum of a sequence allows for determining the period of fundamentals in the sequence. The cepstrum, $\tilde{x}[n]$ is defined in (7). The cepstrum is useful in analysis of composite signals comprised of multiple harmonics since it highlights the frequencies, making it appealing for pitch detection.

$$\tilde{x}[n] = \text{IFFT}\left\{\log\left(\text{FFT}\left\{x[n]\right\}\right)\right\} \tag{7}$$

Pitch is estimated on a frame-by-frame basis, and the range of allowable pitches in a given frame is limited by the typical range of frequencies for male or female speakers. For this work, the frequency ranges used are shown in Table 1. The maximum two peaks in the cepstrum range are chosen, and the 'quefrencies' at these peaks are chosen to estimate the fundamental frequencies/pithces of the frame. Figure 3 highlights this process for a single frame.

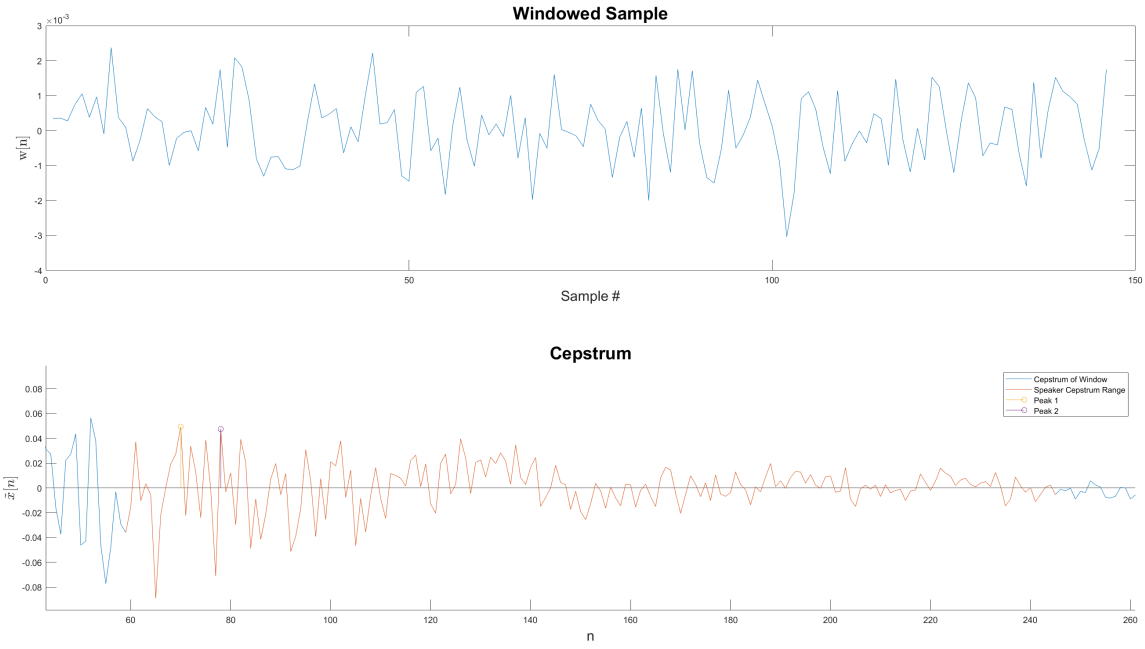

Figure 3: **Top**: Individual windowed frame from audio sample. **Bottom**: corresponding cepstrum labeled to indicate range of estimated pitches based on expected male speaker.

2.2.2 Method #2 Autocorrelation-based Pitch Detection

The autocorrelation function of an audio signal can be used to get fairly accurate pitch estimates. The autocorrelation function (ACF) is defined in (7).

$$r_t(\tau) = \sum_{j=1}^{w-\tau} x_j x_{j+\tau} \tag{8}$$

The autocorrelation is computed frame-by-frame. For the ACF of a frame, the number of samples between the center peak and the next highest peak is its pitch-period estimate. In order to find the second peak, the central peak is suppressed, as are all negative autocorrelation values. The fundamental-frequency estimate or the pitch estimate is simply the sampling frequency of the audio signal divided by the pitch-period estimate. Figure 4 showcases this process for a single frame.

3 Implementation

Given the synthesis and analysis methods outline above, we implemented the LPC vocoder in Matlab. The architecture for this code was inspired by [2], which utilizes autocorrelation for analysis and cepstrum for synthesis (i.e., pitch detection). Our implementation has both of these features as well as autocorrelation-based synthesis. The pseudocode for the implementation of our LPC vocoder can be found below in **Algorithm 1**.

3.1 Frequency Range

3.2 Parameters

The GUI allows for selection of parameters used in reconstruction of signal, but qualitatively, the default parameters enumerated in Table 2 yielded reasonably good sounding approximations of the original audio signals.

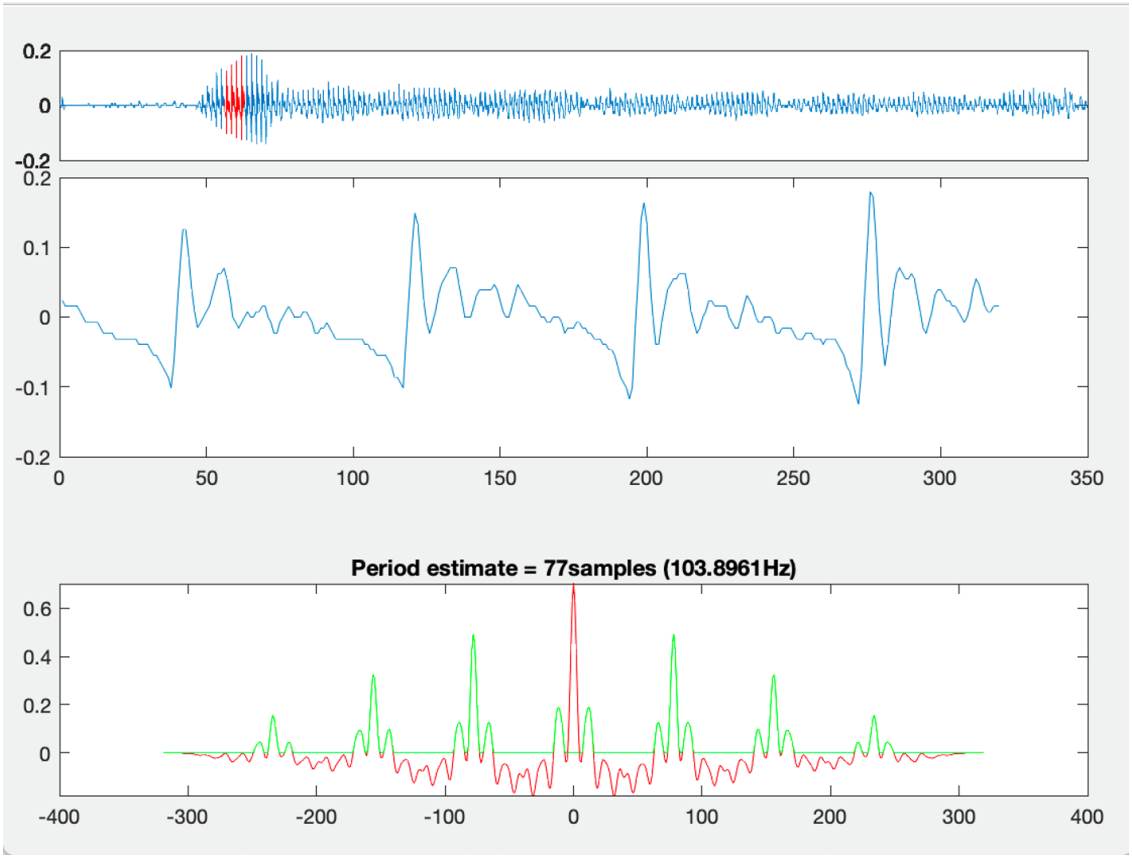

Figure 4: Example of autocorrelation-based pitch estimation.

input : Audio signal: audIn
input : LPC parameters: frameWidth, frameOffset, lpcOrder, detectMethod
output: Synthesized audio signal: audOut

A_k, G_k ⟵ `lpcCoeff`(audIn,lpcOrder);
if detectMethod $==$ *0* **then**
  pitches ⟵ `pitchCepstrum`(audIn,frameWidth,frameOffset)
else
  pitches ⟵ `pitchAutocor`(audIn,frameWidth,frameOffset)
end
pitches ⟵ `smoothPitches`(pitches);
pitches ⟵ `medianFilter`(pitches);
excitations ⟵ `excitationGenerator`(pitches,frameWidth,frameOffset);
audOut ⟵ `synthesizeAudio`(A_k,G_k,excitations);

Algorithm 1: LPC Vocoder Pseudocode

Speaker Class	Low Frequency	High Frequency
Male	60 Hz	250 Hz
Female	150 Hz	350 Hz
Wide Range	50 Hz	500 Hz

Table 1: Fundamental frequency ranges used to limit estimates made by different pitch detection methods.

Parameter	Default Value	Description
Frame Width	60	Width (ms) of analysis frame
Frame Offset	20	Time (ms) by which analysis frame is shifted
LPC Order	20	Number of poles in $A_k(z)$

Table 2: Parameters of interest in LPC Vocoder. Values are adjustable in the GUI.

4 Results

Figure 5 illustrates the output (middle and bottom) of our LPC vocoder based on a particular input (top). Qualitatively, we see that the audio produced by the cepstrum method tends to vary around 0 more consistently than the autocorrelation method, and this is reflected in listening experience, as the cepstrum-based output tends to sound less robotic than the auto-correlation output.

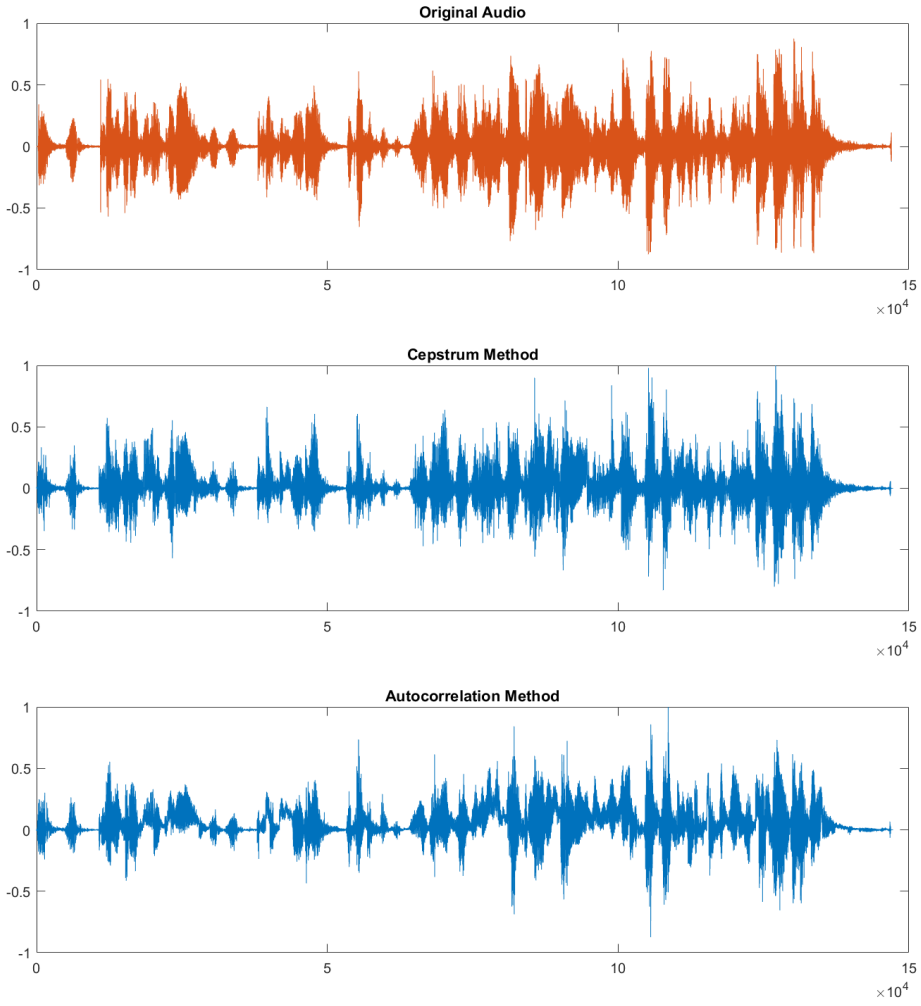

Figure 5: **Top**: Original audio signal from an interview of Bernie Sanders by Killer Mike [4]. **Middle**: Synthesized audio using cepstrum-based pitch detection. **Bottom**: Synthesized audio using autocorrelation-based pitch detection. Qualitatively, the cepstrum-based approximation seems to capture a wider range of pitches than the autocorrelation-based approximation.

5 Appendix

5.1 Code

All code used in this project is available at https://github.com/mdelrosa/eec201_final_project.

6 References

Gutierrez-Osuna, R. (2011, February). *L7: Linear prediction of speech.* Retrieved from http://research.cs.tamu.edu/prism/lectures/sp/l7.pdf

(2015, July 3). Retrieved from https://www.mathworks.com/matlabcentral/fileexchange/45321-lpc-vocoder

Ladefoged, P., & Maddieson, I. (1996). *The sounds of the world's languages.* Oxford.

Mike, K. (2015, December 15). *Talking shop w/ bernie sanders 1/6: Economic freedom — killer mike.* Retrieved from https://www.youtube.com/embed/LCnrQZbqIQU?start=50&end=60

O'Shaughnessy, D. (1988, February). Linear predictive coding. *IEEE Potentials*, *7*.