EE 473 Final Project

Implementation of LPC Vocoder

Egemen Erbayat

15.01.2022

BOĞAZİÇİ UNIVERSITY
ELECTRICAL & ELECTRONICS ENGINEERING DEPT.

Contents

1 Introduction **2**

1.1 Linear Prediction Model for Speech Production . 2

1.1.1 Solution for $\alpha'_k s$ 2

1.2 What is pitch? 2

2 Implementation **3**

2.1 Pitch Detection and Calculation 3

2.1.1 Zero-Crossing Algorithm 3

2.2 Voiced & Unvoiced Decision 3

2.3 Linear Predictive Coding with Gain 3

2.3.1 Gain Computation 4

2.3.2 $a'_k s$ Computation 4

2.4 Synthesizer 5

2.5 Diagram 5

3 Results **5**

3.1 Noise Reduction 5

3.2 Waveforms 6

3.3 Analysis 6

4 References **7**

1 Introduction

1.1 Linear Prediction Model for Speech Production

The fundamental aim is to represent the speech signal with fewer required bits in speech coding techniques. There must not be crucial degradation in the output speech during coding. Speech coding is applied on mobile phones and voice-over IP (VoIP). Linear Prediction Coding (LPC) is the most widely used speech coding in these fields because it does not require a training set. The simple idea of this coding is the current speech sample is approximately equal to a linear combination of past samples[1,2].

$$s(n) = \sum_{k=1}^{p} \alpha_k s(n-k) \text{ for some values of } p, \alpha_k' s$$

where $\alpha_k' s$ are filter coefficients and p is filter order

$\alpha_k' s$ are calculated by minimizing the SSE between the predicted speech and original speech over a finite interval. The SSE error is:

$$E_{\hat{n}} = \sum_m e_{\hat{n}}^2(m) = \sum_m \left(s_{\hat{n}}(m) - \tilde{s}_{\hat{n}}(m)\right)^2 = \sum_m s_{\hat{n}}^2(m) - \sum_{k=1}^{p} \alpha_k \sum_m s_{\hat{n}}(m) s_{\hat{n}}(m-k)$$

1.1.1 Solution for $\alpha_k' s$

Different methods can give different solutions for $\alpha_k' s$. There are mainly two methods for solution:

1. Autocorrelation Method: The signal is windowed to reduce discontinuities using windows that taper the segment to zero, such as Hamming, because the error value will be relatively high at the edges if the signal is not windowed. Then autocorrelation is calculated.

$$\phi_{\hat{n}}(i,k) = \sum_{m=0}^{L-1+p} s_{\hat{n}}(m-i) s_{\hat{n}}(m-k), \quad 1 \le i \le p, 0 \le k \le p$$

$$\phi_{\hat{n}}(i,k) = R_{\hat{n}}(|i-k|), 1 \le i \le p, \; 0 \le k \le p$$

After that the SSE error becomes a Toeplitz matrix:

$$E_{\hat{n}} = \phi_{\hat{n}}(0,0) - \sum_{k=1}^{p} \alpha_k \phi_{\hat{n}}(0,k) = R_{\hat{n}}(0) - \sum_{k=1}^{p} \alpha_k R_{\hat{n}}(k)$$

To find optimal coefficients, Levinson-Durbin Algorithm can be used[3].

2. Covariance Method: The signal is extended, so it does not require to be windowed. The resulting error matrix is symmetric but not Toeplitz. To find optimal coefficients, Cholesky Decomposition Algorithm can be used[4].

1.2 What is pitch?

Pitch is one of the main properties and acoustic sign of sounds to assign musical tones. Pitches are usually associated with frequencies. Pitch information is used in voiced/unvoiced detection and synthesizer of LPC vocoder.

2 Implementation

First, the sound is split into frames using 25 ms Hamming window. Then, the process can be evaluated in four parts.

2.1 Pitch Detection and Calculation

To detect pitch, I used the autocorrelation method firstly. However, I did not get a result as expected when I compared it to the zero-crossing method. Therefore, I decided to use zero-crossing because its computational cost is lower[5].

2.1.1 Zero-Crossing Algorithm

The algorithm can be summarized in five steps:

1. Find indices where the sign of a mathematical function changes.
2. Estimate where zero value is using these indices.
3. Take differences of zero locations. These differences are pitch periods in samples.
4. Reciprocal of pitch period is pitch frequency in samples. To convert it in time, multiply the frequency with the sampling rate.
5. Use mean value for better estimation.

2.2 Voiced & Unvoiced Decision

The decision is made using pitch frequency. If the pitch frequency is in the range of thresholds, the sound is voiced; otherwise, it is unvoiced. The minimum threshold is chosen as 30 Hz, and the maximum threshold is chosen as 500 Hz.

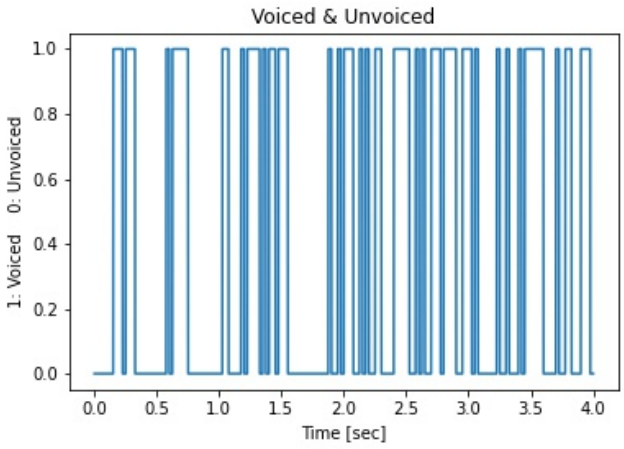

Figure 1: Example Voiced & Unvoiced Plot

2.3 Linear Predictive Coding with Gain

Gain of excitation can be used to matched energy of estimated signal with original signal[6]. When gain is considered, linear combination formula becomes:

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + Gu(n)$$

G is gain and u(n) is impulse train.

2.3.1 Gain Computation

Using the equation above,

$$Gu(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k)$$

And prediction error is,

$$e(n) = s(n) - \sum_{k=1}^{p} \alpha_k s(n-k)$$

$\alpha_k = a_k \Rightarrow e(n) = Gu(n)$, but $\alpha_k = a_k$ is not satisfied always. Therefore, energy is used in gain calculation.

$$G^2 \sum_{n=0}^{L-1+p} u^2(n) = \sum_{n=0}^{L-1+p} e^2(n) = E_{\hat{n}}, \text{ where L is frame length}$$

Gain calculation is not the same for voiced and unvoiced speech. For unvoiced speech:

$$G = \sqrt{\frac{\sum_{n=0}^{L-1} e^2(n)}{L}}, \text{ where L is frame length}$$

For voiced speech:

$$G = \sqrt{\frac{T_{Pitch} \times \sum_{n=0}^{\hat{L}-1} e^2(n)}{\hat{L}}}, \text{ where } \hat{L} = \left\lfloor \frac{L}{T_{Pitch}} \right\rfloor \times T_{Pitch}$$

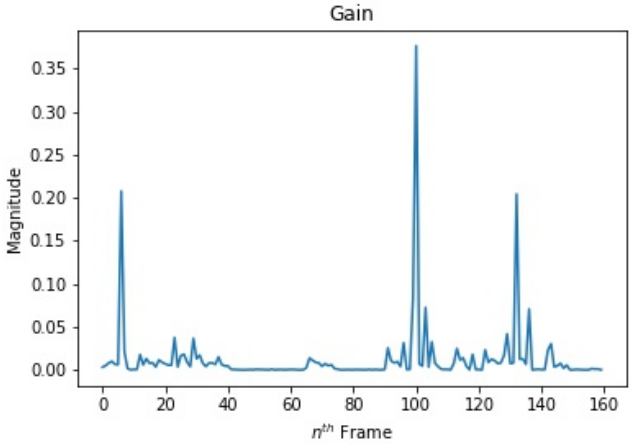

Figure 2: Example Gain Plot

2.3.2 $a_k's$ Computation

To compute $a_k's$, I used a method similar to the autocorrelation method and Levinson-Durbin. Using pseudo-inverse, I found coefficients with minimum error. First, the reverse order of the signal is created. Assume it is A, target signal is S and coefficient matrix is c. We need to solve $A \cdot c = S$. First, take pseudo-inverse of A. $A^{\dagger} = (A^T \cdot A)^{-1} \cdot A^T$. Then, the equation becomes $A^{\dagger} \cdot S = \hat{c}$. S and $A^{\dagger}$ are known. Therefore, $\hat{c}$ can be easily computed.[7].

2.4 Synthesizer

$u(n)$ is a random noise sequence with zero mean and unit variance for unvoiced speech and impulse train for voiced speech.

$$u_{unvoiced}(n) = \mathcal{N}(0,\, 1) \tag{1}$$

$$u_{voiced}(n) = \begin{cases} 1, & \text{if } n = kT_{Pitch},\ k = 0, 1, 2, \ldots \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The transfer function of the synthesizer is

$$H(z) = \frac{S(z)}{GU(z)} = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}}$$

After this filter is applied, decoded speech is obtained.

2.5 Diagram

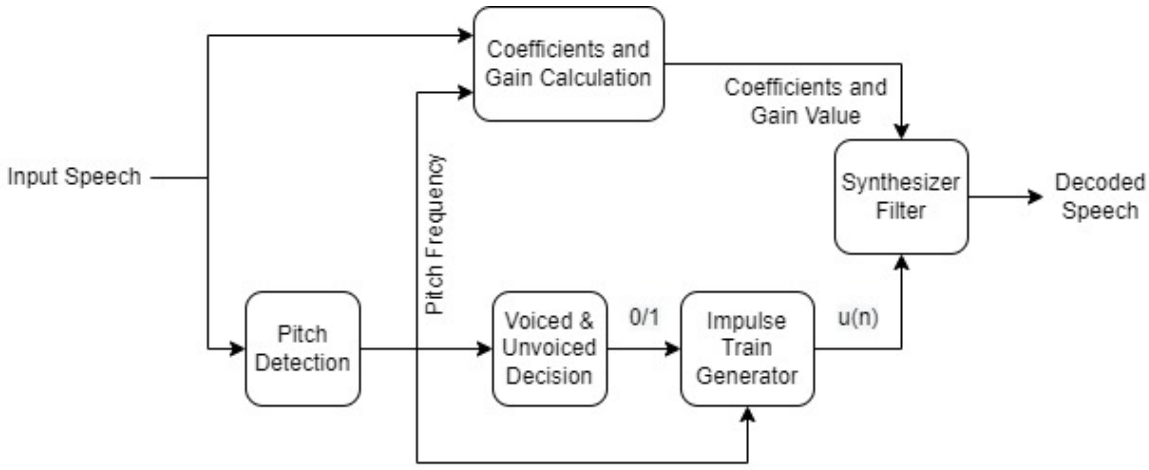

Figure 3: The overall LPC vocoder block diagram.

3 Results

I used python and python notebook to implement the algorithm. The algorithm was tested with different parameters and sounds. The best result I can find has been obtained by adjusting parameters. That is, a better result still can be obtained.

3.1 Noise Reduction

To get better results, the input speech can be pre-processed. One possible pre-processing method is noise reduction. In EE 473 class, we have implemented a noise reduction algorithm as a term project. I have used the spectral subtraction method implemented by my group for noise reduction[8].

3.2 Waveforms

The sound is a part of Ataturk's speech in the 10^{th} anniversary of the Turkish Republic. The filter order has been chosen as 64. Frame length is 25 ms.

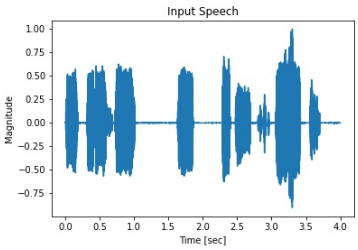

Figure 4: Input Speech

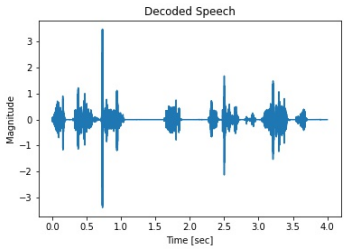

Figure 5: Output Speech

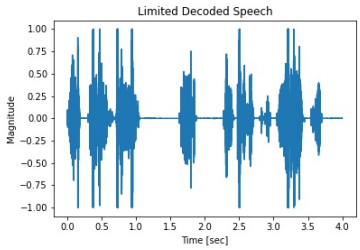

Figure 6: Limited Output Speech

3.3 Analysis

The decoded waveform has unwanted peaks. To remove unwanted peaks, some post-processing algorithms can be used. I did not use any peak removal algorithm. I have just limited the y axis in the range $[-1, 1]$ as a post-process. Waveforms are similar, so it can be seen that the algorithm works well. In addition, when output speech is listened to, significant degradation is not observed. To increase the quality of decoded speech, peak removal algorithms can be used. If any peak removal algorithm is applied, waveforms become more similar.

Overall, a speech can be coded with far less information without degradation and performance loss using LPC vocoder. It can be thought as a data compression process.

4 References

[1] Saranya A., Sripriya N. (2011) LPC VOCODER Using Instants of Significant Excitation and Pole Focusing. In: Nagamalai D., Renault E., Dhanuskodi M. (eds) Advances in Parallel Distributed Computing. PDCTA 2011. Communications in Computer and Information Science, vol 203. Springer, Berlin, Heidelberg.

[2] Deller Jr, John R., John G. Proakis, and John H. Hansen. Discrete time processing of speech signals. Prentice Hall PTR, 1993.

[3] Castiglioni, Paolo. (2005). Levinson–Durbin Algorithm.

[4] Cholesky decomposition. Oxford Reference.

[5] R. G. Amado and J. V. Filho, "Pitch detection algorithms based on zero-cross rate and auto-correlation function for musical notes," 2008 International Conference on Audio, Language and Image Processing, 2008, pp. 449-454

[6] UCSB ECE. DSP Lecture Materials
https://web.ece.ucsb.edu/Faculty/Rabiner/ece259/digital%20speech%20processing%20course/lectures_new/Lecture%2013_winter_2012_6tp.pdf

[7] Technische Universität Ilmenau - Applied Media Systems Group Youtube Channel. \https://www.youtube.com/watch?v=DIr6SPdK4NA&t=79s

[8] A. U. Kaypak, M. Basturk, E. Erbayat, Implementation of Speech Enhancement Algorithms, 2022