520.445

Audio Signal Processing

Fall 2015

PROJECT 2

(Assigned 10/22/15, Due 11/12/15 at 11:59pm)

Project goals:

In this project, you will build an LPC Vocoder based on an all-pole model of speech production. Specifically, you are asked to write a program to estimate the prediction parameters a_k and gain G for the speech signal on a frame-by-frame basis (you can use the MATLAB function *lpc* to estimate the a_k coefficients). Using the prediction data (a_k and G) together with the pitch data provided, the task is to synthesize an approximate version of the sentence. Save your resulting synthesis in a file called *speech_synthetic.wav* (you can use the MATLAB function *wavwrite*). The source signal is the utterance "Calcium makes bones and teeth strong" (found on the course's website). A file called pitches.mat is also provided, which contains pitch estimates every 10 msec.

The ultimate goal of this project is to achieve a compromise between lowest number of LP coefficients and quality of the resynthesized speech. Your final deliverable is a reconstructed waveform with the *best* possible quality, using the *least* amount of coefficients (i.e. filter coefficients, voicing information, pitch estimates, etc). Give a rough estimate in bits/second or samples/second of how much information is required *on average* to reconstruct your signal. Your performance in this project will be assessed in comparison with how well the others perform.

Files provided:

1. The wavfile is a female speech utterance sampled at 16000 samples/sec. Use the MATLAB function *wavread* to load the signal.

2. The file pitches.mat contains estimates of voiced/unvoiced decision and pitch frequencies at a frame rate of 100 frames/sec. A zero value indicates that the speech frame is unvoiced or silence, and a non-zero value is the pitch frequency estimate (in Hz) when the frame is voiced. Note that these estimates are obtained from a correlation analysis of the signal. They are only approximations, and may not be accurate for every frame. Use the MATLAB function *load* to load the pitch estimates.

Few pointers ...

1. Work on this project individually.
2. Pay close attention to your choice of frame rate, as well as number of coefficients to be used. Explore varying the number of coefficients for voiced vs. unvoiced frames.
3. A number of factors may affect the quality of your synthesized speech. For instance, what goes on in a given frame is not independent of what happened in previous frames. As the pitch period changes, you will need to know where the last pitch impulse occurred in the previous frame so as to determine the location of the next impulse in the current frame.
4. You can change the vocal tract filter once per frame, or you can interpolate between frames and change it more often (e.g., at each new pitch period).
5. Listen to your synthesized speech and see if you can isolate what are the main sources of distortion.

Report

Write a report describing how you programmed the LPC Vocoder (e.g., decisions made on frame length, excitation generation, frame overlaps, etc.), along with any graphics. Send your report, all MATLAB code and the synthesized signal to your course assistant <abellur1@jhu.edu>.

Bonus points:

- **Voicing (15 points):** Implement your own voicing detector, which will indicate whether a speech segment is voiced, unvoiced or silence. You can choose a set of parameters, compare their distributions (under each voicing category), and choose a threshold to detect the voicing state. You can use this detector for your LPC project. Your deliverable is a *stand-alone* MATLAB function called *voicingdetector.m*, which takes as input arguments a sound vector, sampling rate, frame length and frame overlap, and returns an array of voicing decisions for each frame with 0 indicating silence, +1 indicating voiced frame, and -1 indicating unvoiced frame.
- **Pitch (30 points):** Implement your own pitch detector based on either the signal's autocorrelation, the LPC residual signal, or any other method and use its output instead of the provided pitch file (pitches.mat). Experiment with the pitch detection method as well as how often you estimate pitch (i.e. frame rate). Does it improve the quality of your reconstructed signal? Again, your deliverable is a *stand-alone* MATLAB function called *pitchdetector.m*, which takes as input arguments a sound vector, sampling rate, frame length and frame overlap, and returns an array of pitch values for each frame and zero for unvoiced and silence frames. Obviously, there is a voicing detection implied in this process.