PART 1: Design a Speech Communication Channel using the LPC Compression Scheme

Implementation

Parameters

The LPC compression and synthesis scheme was implemented with adjustable parameters *L* (frame length), *hop* (frame step or hop size), *zcross_th* (zero-crossing rate threshold used in the voice detector), *rms_th* (root-mean-square threshold used in the voice detector), *order_v* (LPC order for voiced parts), *order_n* (LPC order for unvoiced parts), and *source_type* (excitation signal selection switch, 0 for the impulse train and 1 for glottal pulses).

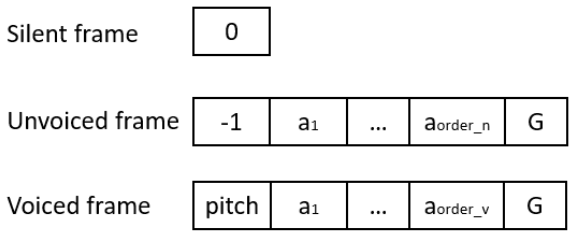
Procedure

Compression

Compression is done by extracting the following information from the input signal with frame-by-frame analysis and storing them in a single vector *comp*:

1) Voice/unvoiced/silent detection result
2) Pitch of the frame in Hz
3) The prediction parameters, coefficients $a_1 \ldots a_p$, computed by Matlab function LPC
4) The gain, which is the square root of the LPC function's prediction error variance

After framing the input signal using parameters *L* and *hop*, a voice detector is applied first to check if a frame is silent, voiced, or unvoiced. If it is silent, the number 0 will be appended to the end of *comp*. If it is unvoiced, the number -1 will be appended, then prediction parameters $a_1 \ldots a_{order\_n}$ and gain G (derived by LPC) will be appended as well. If it is voiced, a pitch detector will be applied to estimate the pitch of the current frame. If pitch estimation is not successful, a pitch value from the nearby frames will be copied instead. Only when no pitch can be found within a 200ms range of the current frame will the frame be changed to unvoiced. If a pitch is found, the pitch value along with the prediction parameters $a_1 \ldots a_{order\_v}$ and gain G (derived by LPC) will be appended to the end of *comp*. After appending all the corresponding information for the current frame, the next frame will be checked. This is repeated until all signal frames are analyzed.



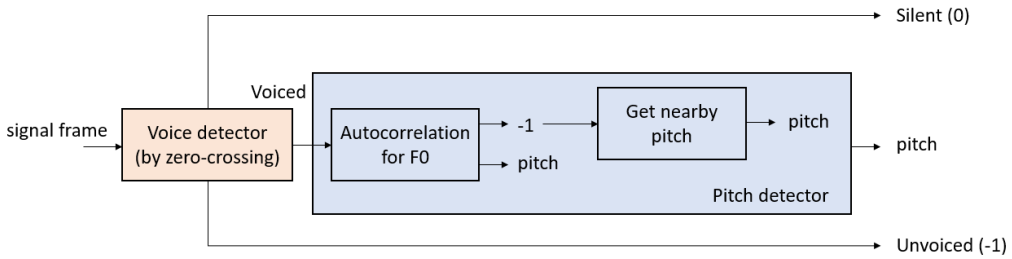
The information storing method for each frame is shown in the figure on the right. Because for this project I will be computing samples/sec instead of bits/sec as the compression evaluation metric, rather than using the standard 1-bit voiced/unvoiced switch at the start of each frame, information of whether the current frame is silent or not is also stored. When all elements in *comp* are treated as samples with the same size, it doesn't matter if the first sample of each frame's compressed info in *comp* is 1 bit or 16 bits. Thus, here the start of each frame in *comp* can be either 0, -1, or a pitch value. By doing this, a lot of space can be saved, as silent frames will require only 1 sample of information. In addition, since we are using a fixed *order_v* for voiced frames and a fixed *order_n* for unvoiced frames, lengths of the three types of frames are all predefined without the need to further indicate the number of coefficients used.

Voice detector

The voice detector works by first calculating the root mean square of the current signal frame. If the computed root mean square value < *rms_th*, flag this frame as silent. If the current frame is not silent, the zero-crossing rate will be computed next. If the computed zero-crossing rate value > *zcross_th*, indicate this frame as unvoiced, otherwise voiced.

Pitch detector

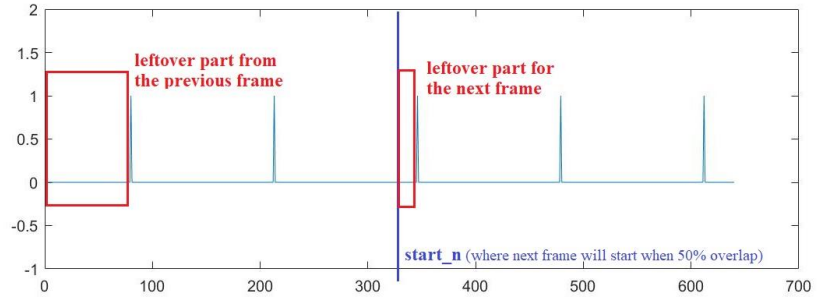
The pitch detector only estimates frames that are flagged as "voiced" by the voice detector. Estimation is done by autocorrelation. Specifically, the period *T0* is derived by the distance between the two largest peaks in the signal frame's autocorrelation waveform, and pitch = 1/*T0*. If either of the following cases is encountered, the derived pitch will be discarded and replaced with a pitch from an adjacent frame (the closest frame within a 200ms range that has a nonzero pitch). 1) Cannot get two peaks in the autocorrelation curve. 2) The derived pitch is not within the normal human pitch range of 50~500 Hz.



Synthesis

Synthesis is done by overlap-adding all the synthesized frames using the same *L* and *hop*. A Hann window is applied on each frame before the frames are added together and normalized to form the final received signal.

Given the compressed info *comp*, firstly the first element in there is checked. If it is 0, the current frame will be a vector of zeros; if it is -1, the current frame will be random noise filtered by coefficients stored in the next *order_n* samples in *comp*, and scaled with the gain stored after the coefficients; if it is a number > 0, the number is used as the pitch to generate a source signal. The current frame will be the source signal (if *source_type* = 0 an impulse train, and if = 1 glottal pulses) filtered by coefficients stored in the next *order_v* samples in *comp*, and scaled with the gain stored after the coefficients. The source signal is circular shifted in a way that its phase is aligned with the source signal from the previous frame (done by shifting the newly generated source signal by the amount of the leftover part from the previous voiced frame and saving the current leftover part amount for the future voiced frame, as shown in the impulse train on the left). After finishing synthesizing the current frame, the system will increment the current index in *comp* and go on to the next frame.



Analysis

Methods described above and the final parameters used are based on the analysis points listed below:

Framing

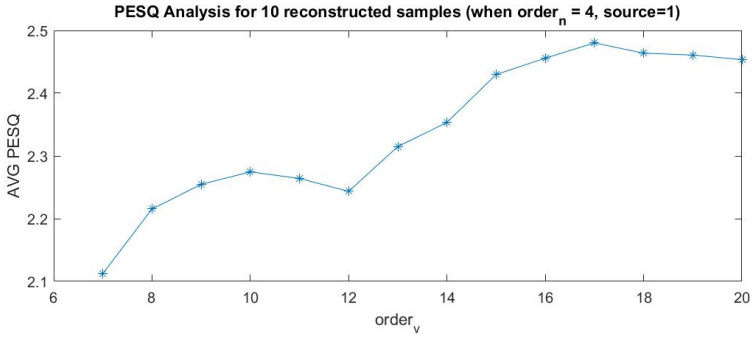
The typical frame length used to do formant analysis is 20~40ms, since that's how long the vocal track stays in a fixed shape. Because a longer frame length will benefit in more precise LPC analysis, 40ms is used here. From testing, it is also found that a lower *L* indeed makes vowels sound a bit shaky.

From testing, it is found that received signals with *hop* = 0.25*L* sound clear, but increasing the step size to 0.5*L* doesn't decrease the sound quality by much. Since using a 50% overlap makes compression much more efficient, *hop* = 0.5*L* is used.

LPC orders

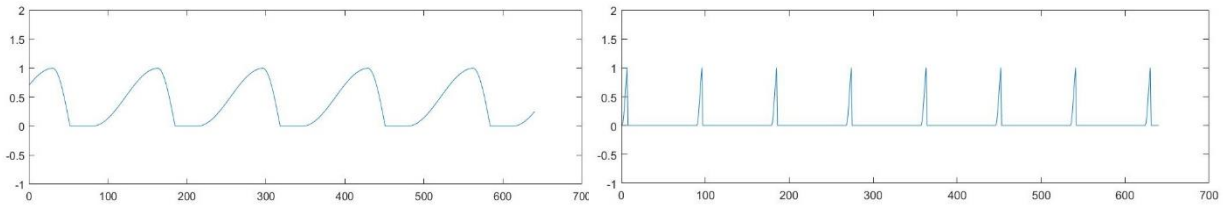
For unvoiced parts, it is found that audio quality does not increase much as *order_n* increases, as their formant structures aren't that complicated compared to those of voiced parts. In the end, an order of 4 is selected to ensure basic quality.

Quality change in voiced parts is more obvious when *order_v* increases. After trying different orders, along with average PESQ tests across all given 10 audio samples, it is found that selecting an order of 10 is intelligible and comfortable enough. Though increasing the order to higher numbers like 17 (refer to figure below) is beneficial, it takes too much space to store the extra coefficients.



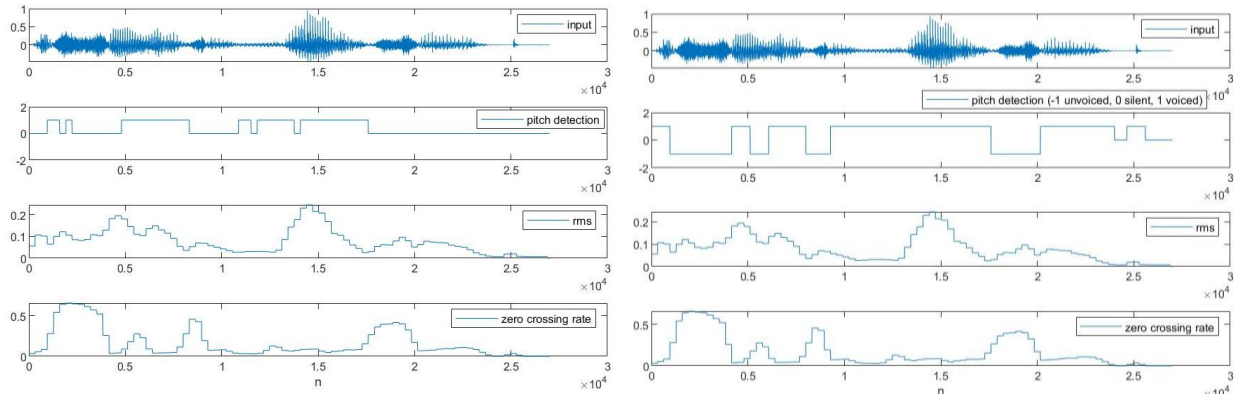
Source generation

It is found that aligning the phases of source signals across different frames (as mentioned in the previous section) greatly increases sound quality. Without this step, there will be distortion. Other than this, the source signal shape also affects the sound quality. An impulse train gets high intelligibility, but sounds mechanic and sharp. Glottal pulses [1] create more natural sounds, but if the pulses are too wide, the sound becomes muffled. In the end, a sharpened version of the Rosenberg pulses (the right figure below) are used.



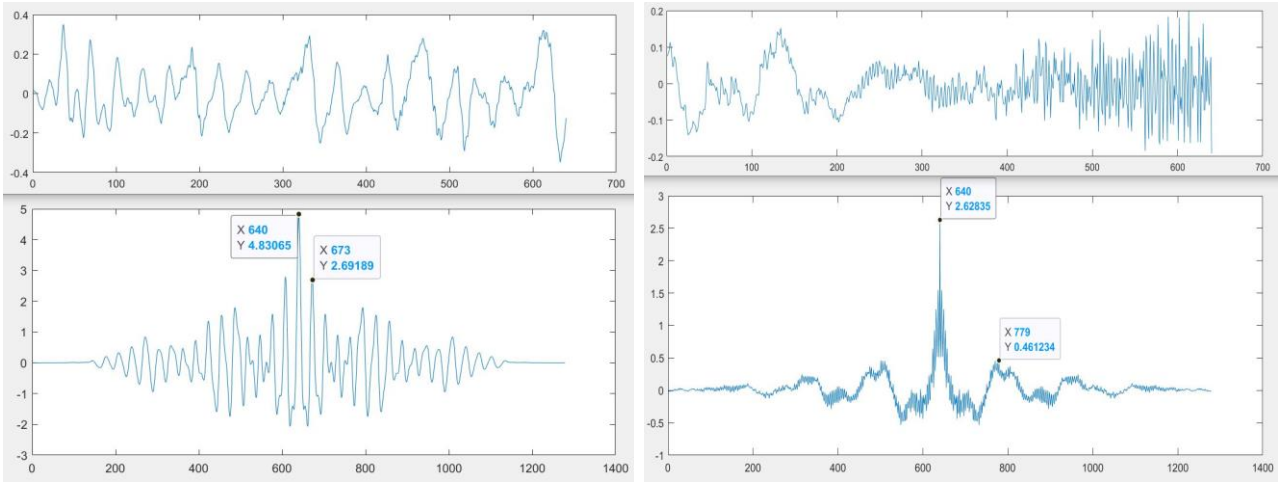
Voice detector

The given mat files sometimes mistakenly classify voiced frames as unvoiced frames, influencing the received signal quality greatly. Since treating unvoiced frames as voiced frames does not actually lower sound intelligibility (but not the other way around), a lower false negative rate with a higher false positive rate for voiced frame detection is actually preferred. The self-implemented voice detector uses rms to identity silent frames, and uses zero-crossing rate to identify unvoiced frames. Since there is a huge difference in the rates between voiced (pitched) and unvoiced (noise with high-frequency changes) parts, detection is very effective, as shown below (old detection on the left, new detection on the right).



Pitch detector

The given pitch mat files also fail to detect lower pitches (< 100 Hz) that contribute to the decline in tones. Autocorrelation is good at estimating pitches, but with some restrictions. For example, for transition frames (frames containing both voiced and unvoiced parts, or frames undergoing changes in pitch), it will be hard to identify the actual periodic peaks in the autocorrelation curve. The figure on the left below detects the second harmonic instead of F0, and it is hard to use simple algorithm to find the right periodic peaks in the figure on the right. By separating voice and pitch detectors, the self-implemented pitch detector is able to compensate the problem. When a pitch cannot be confidently estimated from a frame already identified as voiced, a pitch from nearby frames can be used instead.

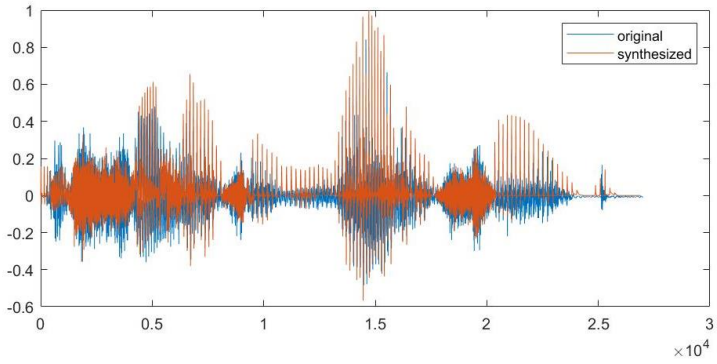


Gain

Without applying the gains derived from LPC, the variation in signal amplitudes lowers speech intelligibility, so gains are stored in the compressed information.

Results

With *L* = 0.04s, *hop* = 0.02s, *zcross_th* = 0.17, *rms_th* = 0.008, *order_v* = 10, *order_n* = 4, and *source_type* = 1, the average PESQ [2] = 2.2853, and the average sample rate = 490.7013 samples/sec. Below is an example result for sample2.



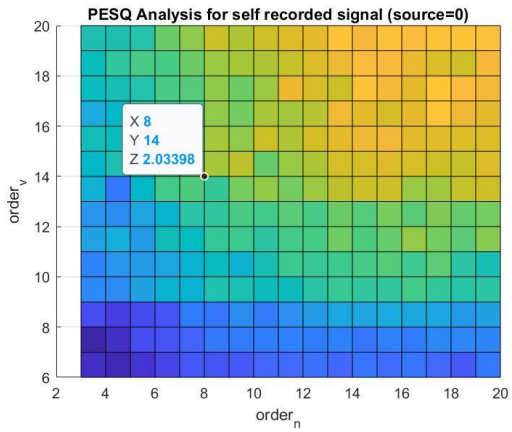
PART 2: Apply the Channel to a Self-recorded Signal

Analysis

Since there are a lot of unvoiced parts in the self-recorded signal, quality of unvoiced parts is actually very important for this specific case. To make sure that almost all unvoiced parts are captured, in this case, *zcross_th* is lowered to 0.12. At the same time, since the percentage of unvoiced parts increases, the average sample rate of compressed information decreases a lot (*order_n* still lower than *order_v*). We can raise *order_v* a bit to achieve even higher speech quality too. According to the figure below, setting *order_v* = 14 and *order_n* = 8 will get an efficient increase in PESQ.

Results

With *L* = 0.04s, *hop* = 0.02s, *zcross_th* = 0.12, *rms_th* = 0.008, *order_v* = 14, *order_n* = 8, and *source_type* = 0 (since I have a higher pitch, the sharp sound output actually feels natural here), the average PESQ = 2.0268, and the average sample rate = 489.6000 samples/sec.



Files

- data\: self-recorded signal and the output audio files
- figs\: figures used in the report
- hw2_LPC.m: the main Matlab file

(PESQ toolbox not included, and input audio samples to be added to data\)

Reference

[1] Montag, M (2014). Speech Production Modeling [Source code]. https://www.mattmontag.com/projects-page/academic/speech

[2] Hu, Y. and Loizou, P. (2008). Evaluation of objective quality measures for speech enhancement. IEEE Transactions on Speech and Audio Processing, 16(1), 229-238.