Lab 2: Framing

Objective of Framing

Any signal can be classified as being either stationary or non-stationary. Speech signals are known to be quasi stationary, i.e. stationary for very short periods of time (10-20 msec, which is the usual frame size that will be used in our program).

Does the 10-20 msec represent the time of a sentence, word or phoneme? Does it make sense that the properties of a signal are stationary throughout a sentence, word or phoneme? Usually people will believe that the 10-20 msec are the timing of a phoneme. However, the properties of a phoneme differ based on what phoneme precedes it and what phoneme succeeds it, i.e. phonemes are not pronounced in isolation. They are co-articulated based on the context in which they exist. Hence, the 10-20 msec, is a time period less than the period of a phoneme, were the properties of a speech signal are usually constant.

Hence, from now on, before processing a speech signal, we divide it into frames of size 10-20 msec, and any processing is done on each frame separately, i.e. preprocessing, feature extraction, pitch period extraction, etc. are all performed on each frame separately.

However, if we take for instance the first frame from 0-10 msec, and the second frame from 10-20 msec, any properties on the boundary between the 2 frames will be lost. Hence, to avoid loss of information/ properties, we usually overlap frames to ensure continuity. The percentage of the frame that overlaps with its preceding frame is called ***"frame overlap"***. The percentage of frame size that is not common between consecutive frames is called ***"frame spacing"*** or ***"frame shift"***. If the amount of frame overlap is x % of the frame size, then the amount of frame spacing/ shift is (1-x)% of the frame size. However, if the frame overlap is defined in terms of milliseconds (e.g. 5 msec), and the frame size is known (e.g. 20 msec), then the frame shift/spacing is 15 msec (20 msec-5 msec).

Performing framing is like multiplying the speech signal by a rectangular window of size 10-20 msec. Multiplying in time domain is equivalent to convolution in the frequency domain. Hence, it is as if the original signals frequency representation is convoluted with the sinc function (rectangular window in the time domain, is a sinc function in the frequency domain), which will distort the signals frequency representation to some degree due to the ripples in the sinc function. The ripples are a result of the sharp fall off at the edge of the rectangular window. Hence to solve this problem after getting the frames, we can multiply each frame by hamming, hanning or any other window type (of the same size as the frame). Such windows have much less ripples in their frequency representation due to their smooth edges in the time domain.

Program Requirements:

Required is a program that divides any input speech signal into frames. The user can specify any frame size, any overlap size and any window type.

Inputs:

1. Audio File Path
2. Frame size in seconds
3. Overlap size in seconds
4. Window type to be used

Outputs:

1. Matrix containing frames (one frame per row)
2. Plot of original signal.
3. Plot of framed signal. This is to ensure that you have framed correctly. If you choose a certain frame size and an overlap size of 50% of the frame size, the framed signal's plot should be almost twice the length of the original signal's plot.

   Note: You can not plot the frames matrix since matrices are 2D and the plot function is used to plot 1D signals. Hence, to make the plot possible, the frames matrix should be used to fill a new vector which will be plotted. The new vector is constructed by concatenating consecutive frames.

Program Summary:

The student uses the "wavread" function to read the audio file into a vector, as well as the sampling rate. Then, the frame size and overlap size (in seconds) can be converted into samples. In other words,

- Frame size (samples) = frame size (seconds) * Sampling Rate (samples/second)
- Overlap size (samples) = Overlap size (seconds) * Sampling Rate (samples/second)

Then the student can iterate on the vector containing the data, and extract samples equivalent to the frame size in samples and with overlap equivalent to overlap size in samples. Such frames are placed in a new matrix, where each row corresponds to a frame. If the user chose a rectangular window, then the previous matrix is the final one. Else, if the user chose a hamming or hanning window, then the student should construct a hamming/hanning window of the same size as the frame size, and multiply each row of the frames matrix by this window. The student should then plot the original and the framed signal.

Sample code:

```
%parameters: frameshift, samplerate, framewidth
shift = round(frameshift * samplerate);
framesize = round(framewidth * samplerate);

nframes = ciel((length(wave) – framesize)/shift );
frames = zeros(nframes,framesize);

for(i=1:nframes)
    frames(i,:) = wave( ((i-1)*shift+1):((i1)*shift+framesize)
       )';
end
```