Fall 2018–2019
Music 320A
Homework #6
Spectrogram, Correllogram
Lab Part 1 of 2[1], Due 11/15/2018 by 11:59pm

Lab Assignments

1. (40 pts) [Short-time Fourier transform] Write a function that generates a spectrogram of an input signal.

```
function stft = myspecgram(x, fs, frameSize, hopSize, fftSize)

% STFT = myspecgram(X, FS, FRAMESIZE, HOPSIZE, DFTSIZE) returns STFT,
% the short-time Fourier transform of the real input signal X (a column)
% using a Hann window FRAMESIZE samples long, skipping HOPSIZE samples
% between frames, and using a DFT length of DFTSIZE.  STFT is a matrix
% that is DFTSIZE tall, with columns containing the DFT of the windowed
% signal segments.
%
% myspecgram() also plots an image of the spectrogram, with the time and
%  frequency axes labeled in seconds and Hz, respectively, according to
% the input sampling rate FS in Hz.
%
% Your Name
```

   Make sure that you use a `DFTSIZE` that is a power of two—the Matlab function `nextpow2()` could come in handy. Also, make sure that `DFTSIZE` is twice the window length.

   Plot the spectrogram image for positive frequencies using the `imagesc` function (see '`help imagesc`'). You will need to use the `axis('xy')` command to have the low frequencies appear on the bottom (rather than the top) of the plot. Plot the spectrogram in dB, with time on the x-axis, frequency on the y-axis, labeling time in seconds, and frequency in Hz. Normalize the spectrogram so that its largest magnitude is 1.0, and use a dB range of `[-60 0]`. Use the command `colorbar` to display a color bar. Try various `colormap()` settings, such as `jet` or ask Elliot for his `Dracula` color map.

   Test your function by generating a 1.0-second-long sinusoid at 1.0 kHz with an amplitude of 1.0 using a sampling rate of 8 kHz. Concatenate this signal with another

[1]Part 2 will be posted soon; it will also be due 11/15/2018.

1.0-second-long sinusoid, this time at 1.2 kHz with an amplitude of 0.2. Add white Gaussian noise with an amplitude of 0.2 (generated using `randn()`) to the concatenated sinusoids. Listen to your signal to make sure that it sounds like you expect. Verify that your spectrogram plot puts the two frequencies where you expect, and that they span the proper times.

Turn in two spectrogram plots, one which very clearly resolves the two frequencies, and another which clearly shows the transition between the two sinusoids. State the signal and analysis parameters you used (*e.g.*, in the figure title—see '`help sprintf`').

2. (30 pts) [Frequency estimation]

   (a) (10 pts) Generate a 1.0-second-long sinusoid in noise signal as above by adding unit-amplitude white Gaussian noise to a unit-amplitude sine wave at $1000/\sqrt{2}$ Hz and using a sampling rate of 48 kHz. Listen to the signal, and verify that you can hear the sinusoid in a background of noise. Form the spectrogram using a window length of 480 samples, and a hop length of 240 samples, and a DFT length of 1024 bins; turn in the spectrogram plot.

   (b) (20 pts) Estimate the sinusoid frequency in each frame of the spectrogram by (a) forming its square magnitude, (b) using `max()` to find the bin k^* at which each frame achieves its maximum value, (c) fitting a parabola to the spectrogram frame square magnitude at the maximizing bin and the two adjacent bins, b^*-1, b^*, b^*+1, and (d) estimating the frequency as that which maximizes the parabola fit to spectrogram frame peak.

   Turn in a plot of the histogram of your estimates using the Matlab function `hist()`, and turn in the mean and standard deviation (the Matlab function `std()` will help) of your estimates in Hz. Was your mean close to the noise-free estimate of about 707 Hz?

   Run your estimator again, this time using a signal generated with noise having an amplitude of 0.25. Turn in your frequency estimate mean and standard deviation for this case. How does the standard deviation of this estimate compare to that of the one above based on the signal with more additive noise?

3. (50 pts) [Frequency tracking, or 'Dave Kerr can whistle'] In this problem, you are going to estimate the frequency trajectory of Dave Kerr's incomparable whistle, captured in the file `Dave_Kerr_whistle_181107B.wav`, and then synthesize

   (a) (5 pts) Read the whistle, and form a spectrogram, selecting the window size to best reveal the frequency trajectory of the Dave whistle fundamental frequency. Turn in a plot of the spectrogram, with the frequency axis ranging from dc to 5 kHz.

   (b) (20 pts) Use the interpolated peak finding method from above to find the frequency and energy of the largest peak in each square magnitude spectrogram

frame. Use the Matlab `hold on` to overlay markers `'.'` at the estimated frequencies for each spectrogram time frame. Don't worry about stray markers plotted for frames when Dave isn't whistling. Turn in your spectrogram plot with markers plotted at the frequency estimates.

(c) (25 pts) Synthesize a signal to match Dave's whistle based on your estimated frequencies and amplitudes. (Hints: Use the Matlab function `interp1()` to interpolate frequency estimates onto a sample-by-sample grid from 0.5 seconds to 6.0 seconds. Do the same for the square magnitudes. Then recall the relationship between sinusoid phase and instantaneous frequency to synthesize your sinusoid.) Turn in a spectrogram plot of your synthesized signal. Briefly describe any differences you hear between the original and your synthesized versions.

4. (50 pts) [Correlogram processing, aka 'Suppress the Professor'] Here, you will process a stereo recording (`sixpack.wav`; don't ask) made with a pair of microphones separated by about five feet, as shown in the figure below. Professors Berners and Abel are standing near the microphones talking. In the first 1.375 seconds, Abel says "Pssst... over here." while Berners is silent. In the next 1.375 seconds, Berners says "Pssst... over here." while Abel is silent. After that, Abel and Berners talk over each other.

   The idea behind the processing is to figure out something about where the professors are standing relative to the microphones, and to produce a pair of monophonic tracks, one in which Abel is suppressed and another in which Berners is suppressed. To do this, you fill form a set of cross-correlations between successive segments of the left and right channels of the stereo recording. The idea is that for any given sound source, the left and right channels roughly should be the same, only differing in amplitude and time delay. Accordingly, the cross correlation is expected to peak at a lag corresponding to the source time delay.

   (a) (20 pts) Use the following commands to read the signal, and filter the signal to the band between 300 Hz an d 8000 Hz so as to remove unwanted low-frequency and high-frequency noise. (Note that the Matlab command `filtfilt()` is used rather than the usual `filter()` so as to produce a zero-phase filter.)

```
% read signal
[signal, fs] = audioread('sixpack.wav');

% design, apply band-pass filter
[b, a] = butter(4, [100 8000]*2/fs);
signal = filtfilt(b, a, signal);
```

   Now, form the cross-correlogram by dividing each channel into overlapping segments `FRAMESIZE` long, and skipping `HOPSIZE` samples between frames. Window each segment channel using a Hann window, and form the cross correlation of the two windowed channel segments using frequency-domain processing. Normalize each segment cross correlation by dividing it by the geometric mean (square

root of the product) of the windowed channel energies. Remember to choose an `FFTSIZE` that is at least twice the size of the frame so as to make room for the non-circular cross-correlation. Note that the `ifft` will return the positive correlation lags at the beginning of the output and the negative lags at the end.

Small hint: A frame size of 2048 samples (almost 50 milliseconds long) and a hop size of, say, 512 samples, should provide good statistical leverage, while being sufficiently short to track changes in time difference of arrival resulting from professor motion.

Make a correlogram image using `imagesc`, displaying the lag (vertical) axis in milliseconds, and the time (horizontal) axis in seconds. Note that since we are looking for time delays between the microphones that were about five feet apart, the cross-correlation is expected to peak in the range $[-5, 5]$ milliseconds. You should then have your correlogram image plot only lags in the $[-5, 5]$ milliseconds range.

(b) (15 pts) Examine the cross-correlation during the "pssst" and "over here" portions of the Abel-only and Berners-only speech to estimate the time delay (in both samples and milliseconds) between the left and right channels for the two speakers. Turn in plots of the overlaid normalized frame cross-correlations (the correlogram slices), four separate plots for the Abel and Berners "pssst" and "over here" sections. This will be about 20 frames per plot. Mark on your plots and the diagram below the estimated time differences of arrival. Explain whether you'd prefer to use the noise-like "pssst" or the voiced "over here" to estimate time difference of arrival between the microphones. Were either Berners or Abel clearly moving while saying "pssst" or "over here"?

(c) (15 pts) Compute the square root of the energy for the two channels during the "pssst" and "over here" sections for Abel and Berners. Delay and scale the left channel signal relative to the right channel signal according to the measured time difference of arrival and channel amplitudes for Abel so that the processed left and right channel signals align in time and roughly match in amplitude. The voiced portion, the "over here" part, of the Abel-only section would be a good section to verify whether the signals are aligned in time and roughly match in amplitude. Once you do this, subtract the processed channels to suppress Abel's contribution to the recording. Listen to this difference signal, and compare it to the sum signal. Do you hear less of a contribution from Abel. (We get roughly 10 dB suppression of Abel and about 2.5 dB suppression of Berners during the voiced section.) Do the same for Berners.

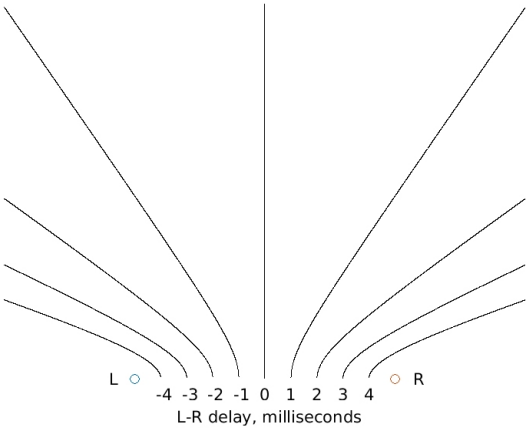

Figure 1: Microphone, time difference of arrival geometry, plan view. Left and right microphone locations are shown along with constant time difference of arrival hyperbolas.