EE 301 Lab 9 – Audio Signals and Systems -- K. Nayak

Introduction

In this lab, you will load, create, and manipulate audio signals in Matlab. Your lab report should be organized by Exercise #. There is also the opportunity to earn extra credit by completing some or all of the "Optional Exercises". If you have other ideas that you'd like to explore for extra credit, contact Krishna via email for approval.

Exercise #1: Warm-Up

Load the clip of my cousin saying "oh my god" using the following commands:

```
[rec fs] = audioread('omg.mp4');        % load
sound(rec,fs);                          % playback
```

The sampling rate of most digital audio is 44.1 kHz. Trim this clip to be exactly one sample and then make both channels exactly the same:

```
rec = rec(1:fs,:);                      % trim
rec(:,2) = rec(:,1);                    % convert to mono
```

Optional: Record your own 1-2 second audio sample or extracting a 1-2 second recognizable audio sample from a song of your choice. There are many built-in Matlab functions that you can use. Helpful links:
http://www.mathworks.com/help/matlab/import_export/record-and-play-audio.html
http://www.mathworks.com/help/matlab/reading-and-writing-files.html

A spectrogram is a tool used to visualize the frequency content of signals as they change over time. It utilizes the Short-Time Fourier Transform (not covered in class), which essentially takes the Fourier transform of short time segments of the complete signal. Generate the spectrogram of the recording and interpret what you see. The horizontal axis represents time and the vertical axis represents frequency.

```
specgram(rec(:,1));                     % display spectrogram
```

Optional: Perform the same operation on a recording of a musical scale. You can record yourself singing a scale or playing a scale on a musical instrument, or download a scale online.

Exercise #2: Artificial Stereo

My dad used to always say "you have two ears and one mouth". My reply was "so do you!" The real reason we have two ears is so that we can detect where sounds are coming from. From an evolutionary perspective, this is important because our ancestors needed to know in which direction to run when they heard a predator.

Most of this information is encoded in time delays. When a sound comes from your left, it reaches your right ear slightly later than it reaches your left ear.

Compute this time delay in seconds? The speed of sound in air is roughly 300 m/s. You will need to measure the width of your head with a ruler.

Now convert this delay into samples (44.1kHz)?

Write a function that creates a stereo signal where one channel is delayed by N samples. You can use the following lines of code within your function:

```
new = rec;                              % copy
rec(:,2) = circshift(rec(:,1),N)        % shift right channel
```

Now put on headphones and test your finding from the previous question. Apply relative shifts to the two channels such that you perceive them as coming from your left, right, or center. Submit these three audio files.

Optional: Use what you have learned to synthesize the panning effect. Record (or download) a longer audio clip, roughly 10 seconds. Write a function that adjusts the relative shift from the start to the finish, so that it sounds like the speaker is moving from one side to the other.

Optional: Try the following experiment. Go to an empty room. Blindfold yourself and have a friend move you to a random place in the room. Then start talking and walking around the room. You may notice that you can sense when you are getting close to a wall. How is this possible?... because humans are capable of SONAR. Simulate this by adding an echo (time shifted version of the signal) to one of the channels. Determine if you can give the listener the sensation that they are close to a wall to their left, right, front, or back.

Exercise #3: Even Better Stereo

Our ears are also sensitive to slight amplitude changes. The relative volume of sound coming to our left and right ears also gives us cues about directionality. Add this to your simulation. Then design and perform an experiment to determine if time delays or amplitude differences are more important in determining the direction that a sound is coming from. You should collect data from at least 3 volunteers.

Open Ended Problems

For those wishing to earn more substantial extra credit, consider one or more of the following open-ended exercises:

1. Automatic Beat Matching. DJ's often need to segue between songs that have a different beat rate. This is the fundamental frequency of the strongest beat (BPM = beats per minute). To generate a smooth transition when the songs' beat's differ, one song is "slowed down" during the transition when both songs are audible.

   Write a function that performs automatic beat matching. Load in two MP4 files that have a different beat rate (BPM = beats per minute). Electronic music is the easiest to start with. Write a function that detects the BPM of each track (Hint: use the Fourier transform!). Write a Matlab function that slows down the faster of the two tracks to produce the smooth transition.

2. Implement an Equalizer in Matlab. Use industry standard cutoff frequencies. Design band-pass filters for each frequency band, scale each band appropriately, and recombine outputs. Make sure each filter has the same group delay. If you want to get fancy, develop a Matlab GUI (graphical user interface) with sliders.

3. Implement one or more of the following audio effects
   a. Realistic Reverb (multiple reflections, different room types)
   b. Flange
   c. Overdrive Distortion
   d. Chorus
   e. Pitch Shifting
   f. Auto-Tune

4. Develop a Matlab model for one or more of the following processes
   a. Audio Compression
   b. A/D + Quantization + D/A conversion