ELEC4400: Perfect Pitch

Thomas Fuller

June 10, 2019

Introduction

Before I learned how money worked or what commerce is, I wanted to be a piano performance major. A lot of the pianists that I looked up to could play things by ear and they had very strong relative pitch or what we commonly know as "perfect pitch"; the ability to hear a note or combination of notes and identify them immediately. I don't have that ability, and vastly overrated my piano abilities in general and "ended up" going to Wentworth instead. But all turned out well, and I ended up learning Fourier Transform and now I basically have perfect pitch too ;]. I guess it takes a little bit longer, but we're gonna listen to some piano chord and ID the notes using Fast Fourier Transform.

Process

1. I've placed some files on the blackboard of someone else playing the piano. Download these.

2. Import the following libraries into your python script.

```
import matplotlib.pyplot as plt

import numpy as np

from scipy.fftpack import fft

from scipy.io import wavfile
```

3. Underneath our imports, we're going to write a function and call it a couple times. The structure of how you write that is `def functionName(input1,input2,...):` and then everything after that function declaration which is indented will be exected each time we call the function. At the bottom of the script, we're going to put `if __name__ == "__main__":`. It functions kind of like the "main" function of a C script. Check out the template script I've put on the blackboard, which I think will clarify how the script should be organized.

What the function should do

1. Call `wavfile.read()` to read in the input audio file. This function outputs two things: the sampling frequency of the wav file in samples per second, and the data corresponding to the sound file. Save the sampling frequency into a variable called "fs". Save the data into a variable called "data".

2. Call `fft()` in order to take the fast fourier transfrom of the data. Save this transformed data to a variable called "fftOut".

3. Find the total amount of time in the recording by evaluating `len(data)/fs`. `len(data)` is the length of the data.

4. Call `np.arange()`, using `len(data)` as its input. The idea here is to generate a list of the sample indices. spanning from zero to however many samples are in this particular recording. Save the list of sample indices to a variable called "n".

5. We've known for a while that $t = nT$, where T is the duration of time between samples. `wavfile.read()` gives us the sampling **frequency**. $T = 1/f$. Use these two equations and also the list object you created in last step to generate a list of the frequencies contained in the recording. Store this list to a variable called "freqLabel".

6. Call `plt.plot()` to plot the absolute value of fftOut vs freqLabel. Title the plot "Piano Note Analysis". The x axis should be labeled "frequency (Hz)", and the y axis should be labeled "amplitude". The limits on the x axis should be from 0 Hz to the frequency of hte highest note on a piano in Hz.

What the overall program should do

1. Call the function you wrote above to take the fast fourier transform of the two audio files, and produce the plots. 2. To figure out what notes are being played, look at the first 3 fundamental frequencies, or the first three points on the plot where the amplitude is way higher.

Questions

1. For the calculation in 3 under what the function should do, analyze the units to convince yourself that what you calculated is the time in seconds.

2. Based on your fourier transform analysis what notes were being played in the recording?

3. I've linked here a video of the notes which are being played. KPT HKP. Did your analysis match what these videos showed?

What you turn in

1. Submit code, plots, and answers to the three questions.