GUITAR TUNER

Tuner Master

Authors: Prakriti Sharma (ps4425), Rishabh Guha (rg4023)
Prof. Ivan W Selesnick

22nd December 2021

OBJECTIVE

In this project, we have created an interactive guitar tuner called the ‘Tuner Master’. We have taken inspiration for this idea from a music educational app called Yousician. As a guitar player, it was a very fascinating idea to create our own tuner that could be used to tune my guitar. The application takes input from the microphone in real-time and determines the frequency of the input sound to identify which musical note it sounds the closest to. The UI changes dynamically to indicate the frequency difference between the input and the closest true note. This way, it will align the measured frequencies with the note names in scale. If the frequencies match a particular note, the tuner will display the names of that note on the screen.

DESCRIPTION

A tuner is a device that measures the frequencies produced by vibrating strings on instruments like a guitar or a ukulele. In our project, we have specifically created a guitar tuner that measures the frequencies produced by the vibrating strings of an electric or acoustic guitar. It then aligns the measured frequencies with notes on a scale (eg. 440 Hz is 69 on the scale in our project). For example, if you play a C# note leaning towards a C note, you will be prompted to either tune down to a C note or tune up to a D note. If the frequencies match a particular note correctly, the tuner displays the name of that frequency on the screen.

For developing this application, we have used python, Tkinter, PyAudio, and NumPy. We used PyAudio to read the data from the microphone which is received as a raw audio stream. We create a buffered queue of limited size with lock implementation to hold the read streams of data. We pop these data elements one by one and apply a fast-fourier-transformation (numpy.fft) on them. Then, an HPS (Harmonic Product Spectrum) operation is performed to filter the harmonic frequencies. From this HPS output, we get the loudest frequency and then we convert it to a musical note.

In our application, we have also created a user interface to enhance the accessibility and efficiency of our tuner. The UI has multiple components. The circle on the center displays the closest true note while the needle indicates the variance of the input signal from the true note. Two text boxes on either side display the closest notes that are a semitone up and down.

THEORY

We have used the following python libraries to develop this application:

- PyAudio: It provides Python bindings for PortAudio v19, the cross-platform C++ audio I/O library. With PyAudio, we can easily use Python to play and record audio on a variety of platforms, such as GNU/Linux, Microsoft Windows, and Apple macOS.
- Tkinter: It is an open-source, portable graphical user interface (GUI) library designed for use in Python scripts. It relies on the Tk library, the GUI library used by Tcl/Tk and Perl, which is in turn implemented in C. Therefore, Tkinter can be said to be implemented using multiple layers.
- NumPy: It is a python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. NumPy provides an array object that is up to 50x faster than traditional python lists.
- Wave: It is the python library that provides an easy interface to the audio WAV format. The functions contained in this module can write audio data in the raw format to a file-like object where we can read the attributes of the WAV file such as sampling rate, number of channels, etc.

PROCEDURE

The process starts with the audio analyzer retrieving the note input using the device microphone. PyAudio is used to read the microphone input. Upon accessing the microphone buffer, fourier transformation is applied to the audio buffer to decompose the musical chord into its constituent frequencies. The padding and Hanning window technique is used to ensure accurate frequency resolution. Post Fourier transformation, we retrieve the frequency of the loudest note. This frequency note is added to a queue for processing.

The steps to process the frequencies from the queue to identify the note can be summarized as follows:

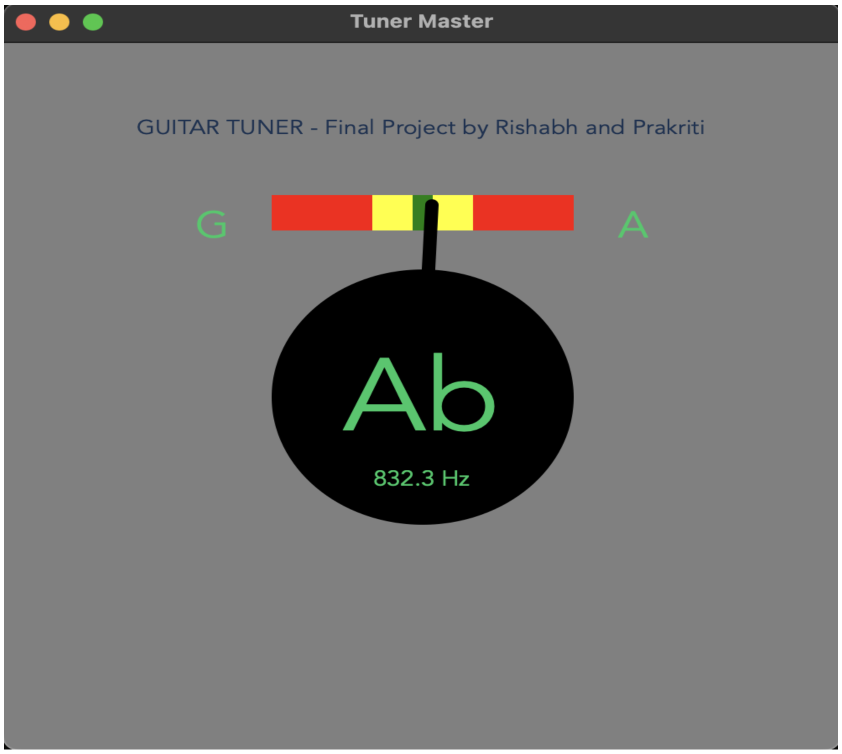
- Once we have obtained the frequency of the loudest note in the input, we can use our knowledge of music theory to calculate the note from this frequency. We have referred [2] in order to get a mapping of the notes on the musical scale and their respective true frequencies.

- We choose A5 as our base note. It has a frequency of 880 Hz and is the 81st note in the musical scale, which starts from A0. Hence we use the formula **12 * np.log2(freq / 880) + 81** where we basically subtract the frequency of A5 from our input and add its note number value i.e. 81. We then multiply this by the number of notes in an octave i.e. 12 to get the numerical number of the note relative to the start of the musical scale i.e. relative to A0.
- We calculate the **result % 12** to get the index of the note from our array where we have the 12 distinct notes stored in order. This will give us the resulting note value to display.

RESULT

We have tested the application on Youtube videos of true note frequencies as well as actual guitar input. The results showed that the tuner was able to predict the correct note with high accuracy.

Final working UI preview :



REFERENCES

[1] https://medium.com/@ianvonseggern/note-recognition-in-python-c2020d0dae24

[2] https://pages.mtu.edu/~suits/notefreqs.html

REPOSITORY LINK

Here is a link to the repository code on GitHub : https://github.com/sharmaprakriti73/DSP-Lab