Interesting Projects To Strenghthen DSP Teaching

Sophie X. Liu, Rohan George Aby, Matthew Samuelson, Tevin Macias, and Emily Garvie
Engineering School, Oral Roberts University, 7777 S. Lewis Ave Tulsa OK 74171, USA

Abstract— Education is moving away from traditional rows of students looking at the same textbook while a teacher lectures from the front of the room. Today's classrooms are evolving to use more technology and digital resources. Students are more interested learning technical material if they can see useful applications for it. When teaching the digital signal processing (DSP) course, it is possible to develop projects to show how the techniques can be used in real world application. This paper presents three simple, yet interesting projects in the author's undergraduate DSP course to motivate the students learning the Fast Fourier Transform (FFT).

Keywords—pattern recognition; DSP; digital guttar tuner, power spectrum density

I. RECOGNIZE HUMAN VOICE

The objective of this project was to write a program in MATLAB that could analyze an .m4a recorded file of a human voice and determine what note the voice was in. It would do this by reading the audio file, then splitting the data into the file's frequency and sample data. The Fast Fourier Transform (FFT) was taken of the sample data and used to plot the graph of the FFT's amplitude verses frequency. The sample frequency from the original file was changed to have the same vector length as the FFT. The waveform is imperfectly periodic, and the frequency of the first period was used to find which note was being sung. A table of the musical note frequencies was used to determine this, as well as whether or not the note sung is sharp or flat.

The program could be implemented as a tone adjuster, or an instrument tuner. If a certain key is desired, the user could repeated change their input sound in order to change the frequency to meet a certain note. In order to make this method more practical and time efficient, the program could further be improvised to have an instantaneous input. This could be a microphone or an instrument plugged into to the computer so that MATLAB could read it. In this way a person would more practically be able to know the note they are inputting. Also, the program could be further revised to provide the change of frequency needed if the note is either flat or sharp. Five examples were recorded of random voice notes and were evaluated in the program. Their frequency responses shown in Fig. 1.1 to 1.5 were graphed and the frequencies were compared with the excel sheet in Table 1 to determine what key they were in.

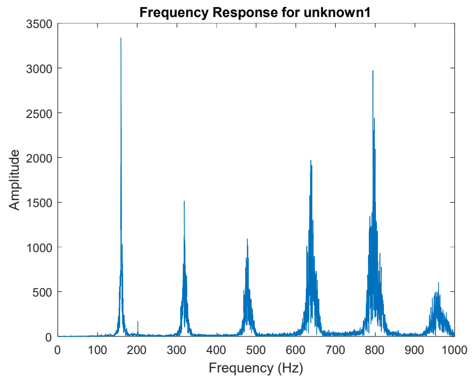

Fig. 1.1 Example 1 Output: The note frequency is 155.2
The key is D# / Eb in octave 3

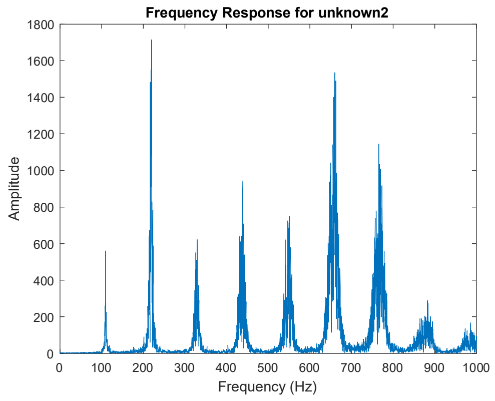

Fig. 1.2 Example 2 Output: The note frequency is 109.5
The key is A in octave 2

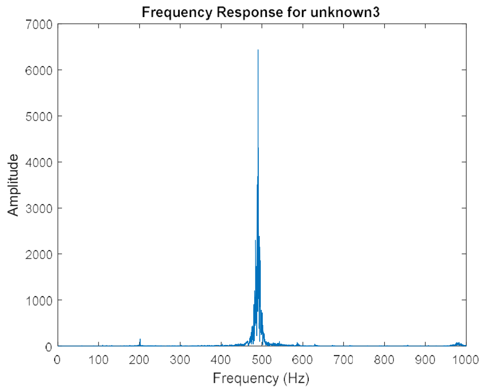

Fig. 1.3 Example 3 Output: The note frequency is 475.3
The key is flat B4 (or sharp A# / Bb4)

TABLE 1.1 MUSICAL NOTE FREQUENCIES AND OCTAVES

C	0	16.35	C	5	523.3
C# / Db	0	17.32	C# / Db	5	554.4
D	0	18.35	D	5	587.3
D# / Eb	0	19.45	D# / Eb	5	622.3
E	0	20.6	E	5	659.3
F	0	21.83	F	5	698.5
F# / Gb	0	23.12	F# / Gb	5	740
G	0	24.5	G	5	784
G# / Ab	0	25.96	G# / Ab	5	830.6
A	0	27.5	A	5	880
A# / Bb	0	29.14	A# / Bb	5	932.3
B	0	30.87	B	5	987.8
C	1	32.7	C	6	1047
C# / Db	1	34.65	C# / Db	6	1109
D	1	36.71	D	6	1175
D# / Eb	1	38.89	D# / Eb	6	1245
E	1	41.2	E	6	1319
F	1	43.65	F	6	1397
F# / Gb	1	46.25	F# / Gb	6	1480
G	1	49	G	6	1568
G# / Ab	1	51.91	G# / Ab	6	1661
A	1	55	A	6	1760
A# / Bb	1	58.27	A# / Bb	6	1865
B	1	61.74	B	6	1976
C	2	65.41	C	7	2093
C# / Db	2	69.3	C# / Db	7	2217
D	2	73.42	D	7	2349
D# / Eb	2	77.78	D# / Eb	7	2489
E	2	82.41	E	7	2637
F	2	87.31	F	7	2794
F# / Gb	2	92.5	F# / Gb	7	2960
G	2	98	G	7	3136
G# / Ab	2	103.8	G# / Ab	7	3322
A	2	110	A	7	3520
A# / Bb	2	116.5	A# / Bb	7	3729
B	2	123.5	B	7	3951
C	3	130.8	C	8	4186
C# / Db	3	138.6	C# / Db	8	4435
D	3	146.8	D	8	4699
D# / Eb	3	155.6	D# / Eb	8	4978
E	3	164.8	E	8	5274
F	3	174.6	F	8	5588
F# / Gb	3	185	F# / Gb	8	5920
G	3	196	G	8	6272
G# / Ab	3	207.7	G# / Ab	8	6645
A	3	220	A	8	7040
A# / Bb	3	233.1	A# / Bb	8	7459
B	3	246.9	B	8	7902
C	4	261.6	C	9	8372
C# / Db	4	277.2	C# / Db	9	8870
D	4	293.7	D	9	9397
D# / Eb	4	311.1	D# / Eb	9	9956
E	4	329.6	E	9	10548
F	4	349.2	F	9	11175
F# / Gb	4	370	F# / Gb	9	11840
G	4	392	G	9	12544
G# / Ab	4	415.3	G# / Ab	9	13290
A	4	440	A	9	14080
A# / Bb	4	466.2	A# / Bb	9	14917
B	4	493.9	B	9	15804

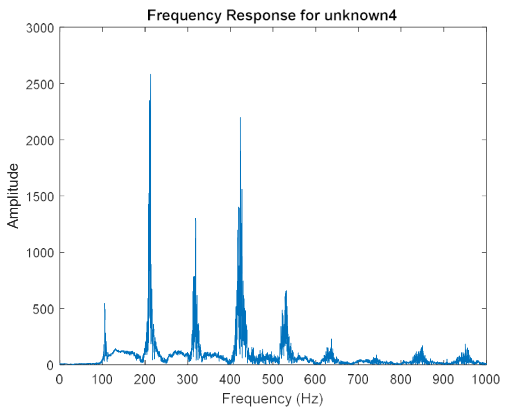

Fig. 1.4 Example 4 Output: The note frequency is 105.3
The key is G# / Ab in octave 2

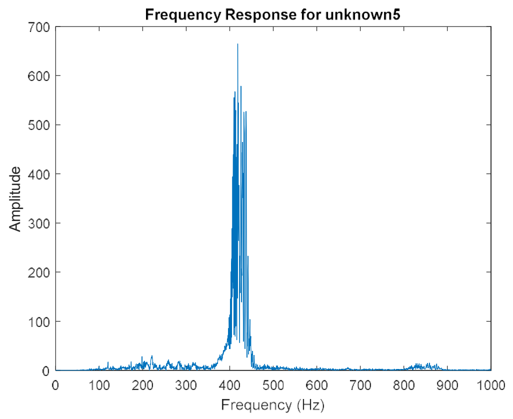

Fig. 1.5 Example 5 Output: The note frequency is 407.8
The note is a flat G# / Ab4 (or sharp G4)

Each example was evaluated using an online virtual keyboard shown in Fig. 1.6 at the following URL: http://piano-player.info/. Each determined note closely sounded like the corresponding note from the virtual keyboard. This shows the program works and can be relied on to find an accurate note reading.

Fig. 1.6 Virtual Keyboard

II. DIGITAL GUTTAR TUNER

A. *Methodology*

A GUI was created using app designer in MATLAB for easy interfacing. The GUI has two button options: one is called "Tune by Ear" and the other is called "Tune by Sight" that is based on the recording. A dropdown menu allows the user to select which note they are attempting to generate or verify tuning with. Two graphs display the frequency data. The computer tone is on top and recorded tone is on bottom. If the "Tune by Ear" option is selected, the computer generates the pitch corresponding to the letter and then the Fast Fourier Transform is applied and a "Amplitude" vs "Frequency" graph is displayed for the computer tone. If the "Tune by Sight" option is selected, the user will play a note on the guitar. The analog frequency of the note is recorded and then the Fast Fourier Transform is applied to the signal. Then an "Amplitude" vs "Frequency" is graphed for both with the user recorded note on the bottom and computer generated note on top. The two graphs can then be compared for peak values.

The computer generated frequencies were layered to the fifth octave to produce a fuller tone and a well-rounded graph. The frequencies were determined in reference to A /49/440Hz. Table 2.1 gives the information that is from *Piano Key Frequencies* on Wikipedia.

TABLE 2.1 PIANO KEY FREQUENCIES

Note	Note Number	Frequency (Hz)
E	32	164.81
A	37	220.00
D	42	293.66
G	47	391.99
B	51	493.88
e	56	659.25

B. *Testing Procedure*

This was tested in two parts. First, the "Tune by Ear" option was selected. Each note was played to make sure the correct pitch was being generated. The duration of the note was adjusted and a sampling frequency of 11,025 Hz was chosen for fast processing time and low resolution audio. Then the "Tune by Sight" was tested. A guitar was connected through a digital audio interface for recording the signal. The FFT was then used to convert this signal to a frequency signal that could be plotted. Once plotted, the graph was compared to the graph of the frequency graph of the computer-generated note. The guitar's tuning was then adjusted accordingly until the two graphs correlated.

C. *Experimental results*

The guitar's signal had more peaks than the computer-generated signal because the guitar string does not generate a pure tone. This, however, did not affect the ability to determine which peak should be compared for tuning. In the graphs below, the recorded frequencies peak at the specified frequency or at an octave of the frequency.

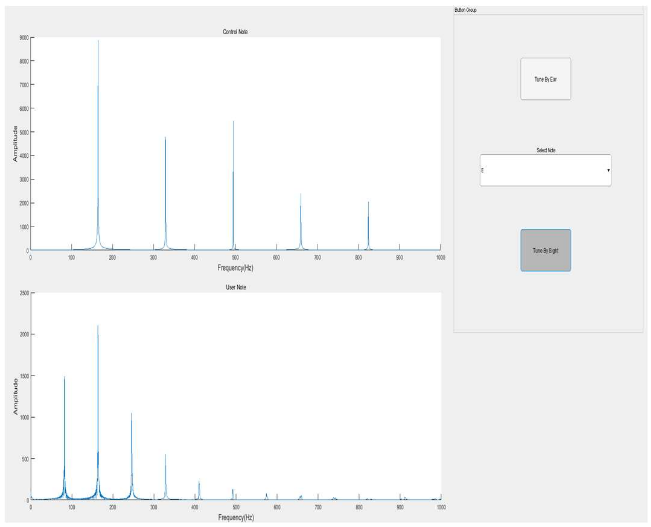

Fig. 2.1 E-Note.

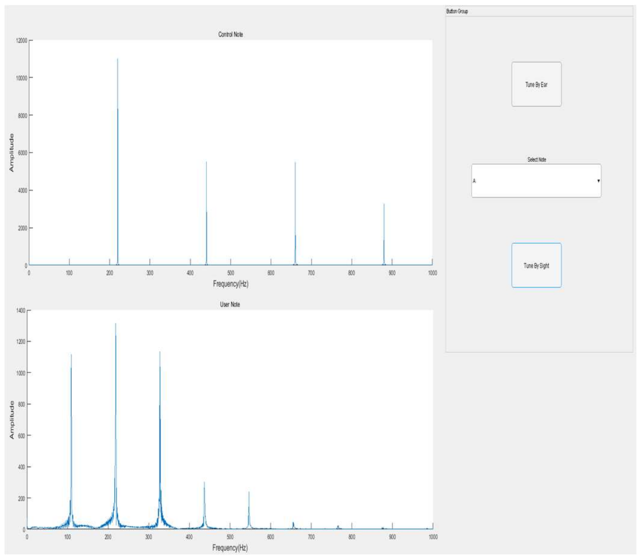

Fig. 2.2 A-Note.

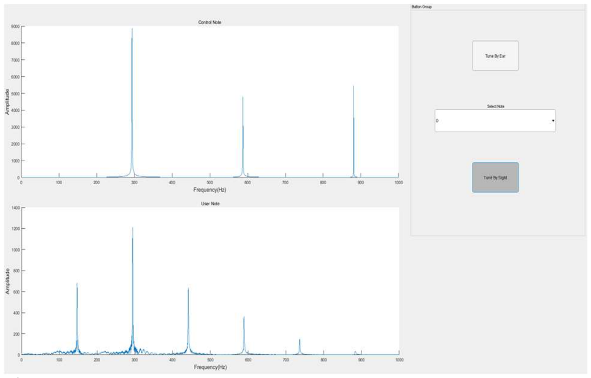

Fig. 2.3 D-Note.

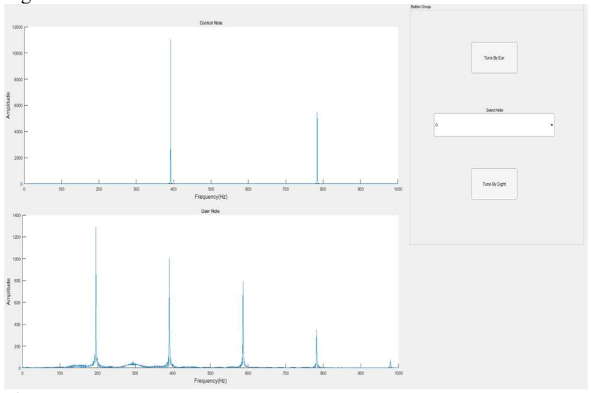

Fig. 2.4 G-Note.

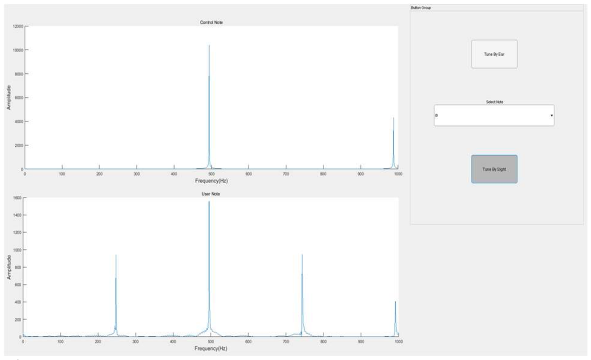

Fig. 2.5 B-Note.

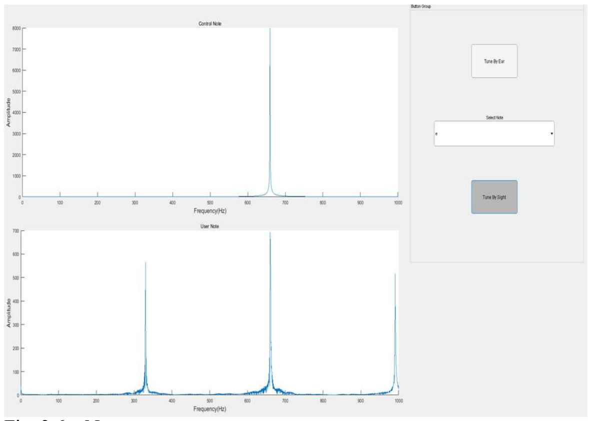

Fig. 2.6 e-Note.

D. Conclusion

In conclusion, this project produced a tuner was capable of generating tones for the EADGBe strings of a guitar, and used the Fast Fourier Transform to change computerized frequency and recorded guitar signal to a frequency graph that can be compared with each other. In the future, the two graphs could be overlapped, there could be a more precise reading of how much adjustment the guitar needs, and it could be made more portable.

III. DEFINE TWO FEATURES

The project is to use simple features and pattern recognition technique to differentiate the male and female voice. The voice signals are digitized and analyzed first. Two features - power spectrum density ratio and top frequencies average are extracted for machine learning and voice recognition.

A. Power spectrum density

The power spectrum describes the distribution of power into frequency components composing that time domain signal. The power spectrum density or power spectral density (PSD) of the signal describes the power present in the signal as a function of frequency, per unit frequency. FFT algorithms are fast algorithms for computing the discrete Fourier transform (DFT). This is achieved by successive decomposition of the N-point DFT into smaller-block DFT taking advantage of periodicity and symmetry. Practically, a PSD is obtained by taking the amplitude of the FFT, multiplying it by its complex conjugate and normalizes it to the frequency bin width. Specifically:

$$|X(f_k)| = \left|\sum_{n=0}^{N-1} x[n] \exp\left(-\frac{j2\pi nk}{N}\right)\right|$$

$$PSD = \frac{|X(f_k)|^2}{N}$$

where $|X(f_k)|$ is the N point FFT magnitude at bin k of a sequence $\{x[0], x[1] \ldots x[N-1]\}$ for $f_k = \frac{k}{N}$ and $k = 0, 1, 2, \ldots N-1$.

After calculating PSDs of both male and female voices, making PSD graphs for two groups, it was observed that generally the male voice has higher power spectrum density. The results are in the Fig. 3.1. The blue signal is the male voice and the brown signal is the female voice.

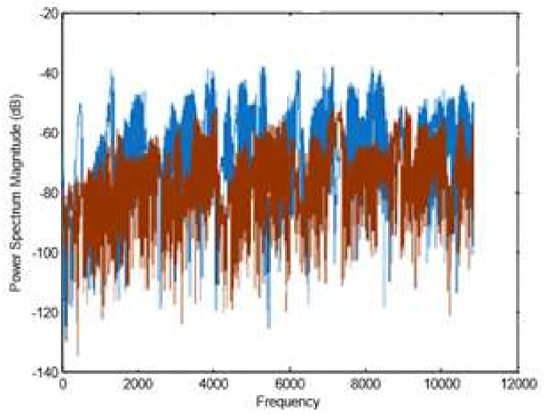

Fig. 3.1 Power spectrum density of the two sexes.

B. *Feaure 1: PSD Ratio- PSDR*

We want to define a feature relevant to PSD independent of individual magnitude of PSD. One would be taking the average of the magnitude of the FFT components that correspond to the high frequencies (0 to 6000Hz) and dividing it by the average of the magnitude of the FFT components that correspond to the low frequencies (6001 to 11000Hz). The feature is denoted as PSDR, a ratio of PSD. Let Mj be the magnitude of PSD at frequency f_j for a voice sample and $j = 0, 1, 2, \dots, \text{mid}, \dots Q$. $f_{mid} = \frac{f_0+f_Q}{2}$. f_{mid} is the median of the frequency range.

$$\text{PSDR} = \frac{\frac{1}{Q-mid}\sum_{j=mid}^{Q} PSD_j}{\frac{1}{mid-1}\sum_{j=0}^{mid-1} PSD_j}$$

C. *Feature 2: Top frequenies average- Top10*

Consider a fact that the highest frequency the voiced speech of a typical adult male can reach is lower than that a typical adult female can do. It would be necessary to add a feature that can distinguish male and female voices through the high range of frequencies. Therefore, we define top frequencies average as follows:

TOP10 = average of top 10% frequencies in a voice

IV. MACHINE LEARNING AND RECOGNITION

We take total N voice samples. Half of them are female voices forming a female group. The another half are male voices forming a male group. In both groups we take 70% samples respectively for machine learning/training forming a training group. The rest samples (30%) for each group are for recognizing/testing the algorithm forming a testing group.

A. *Machine learning*

For each sample in the training group, calculate PSDR and Top10. Calculate the maximum and minimum of PSDR for female group and male group respectively. Calculate the maximum and minimum of Top10 for female and male group respectively. Because this is supervised learning, we can the ranges of each feature for female and male. For example, Top10 in the female group falls in the range of [Top10_min_F, Top10_max_F] and that in the male group falls in [Top10_min_M, Top10_max_M]; PSDR in the female group falls in the range of [PSDR_min_F, PSDR_max_F] and that in the male group falls in [PSDR_min_M, PSDR_max_M].

B. *Recognition processing*

During the recognition process, the table 3.1 is used to distinguish the female and male voices. Only samples in the testing group are involved in this process. For kth sample in the testing group we need calculate its PSDR and Top10. Here we use PSDR_k and Top10_k for the features of kth sample.

If $\text{PSDR\_k} \in [\text{PSDR\_min\_F} \pm \Delta_1, \text{PSDR\_max\_F} \pm \Delta_2] \cap$ $\text{Top10\_k} \in [\text{Top10\_min\_}F \pm \Delta_3, \text{Top10\_max\_}F \pm \Delta_4]$
kth sample -> female voice

If $\text{PSDR\_k} \in [\text{PSDR\_min\_M} \pm \Delta_{11}, \text{PSDR\_max\_M} \pm \Delta_{22}] \cap$ $\text{Top10\_k} \in [\text{Top10\_min\_}M \pm \Delta_{33}, \text{Top10\_max\_}M \pm \Delta_{44}]$
kth sample -> male voice

Otherwise reject, need further distinguish

Where $\Delta_1, \Delta_2, \Delta_3, \Delta_4, \Delta_{11}, \Delta_{22}, \Delta_{33}$, and Δ_{44} are pre-determined parameters based on given knowledge and experience. This project was to acquire voice samples of males and females reading John 3:16, bible verse in the Bible NIV version that were used was 50 and 50 voice samples respectively.

We use two simple features described above achieves a recognition rate of 89% in deciding between female and male voices in the testing samples. It illustrates how the PSD ratio and top frequencies average of an audio signal can be used for differentiating the male and female voice. To increase the recognition rate, more features and more training samples could be included.

REFERENCES

[1] https://en.wikipedia.org/wiki/Piano_key_frequencies