Guitar Tuner Using Correlation (MATLAB Analysis)

PROJECT REPORT

Signals and Systems (ECE1004)

Submitted By:

- Paritosh Pandey(17BEC0308)
- Arjun T(17BEC0191)
- Ayush Kumar Agrawal(17BEC0668)
- Vineeth (17BEC0711)
- Palak Khatri(17BEC0383)
- Sai Aakash Bhatnagar(17BEC0593)

Slot C1 and C2

PROBLEM STATEMENT

Guitar Tuner implemented in Matlab and used to tune guitar for given different strings. Or develop a precision instrument tuner that would be able to determine within a finite range perfect intonation.

INTRODUCTION

Speech-recognition technology is embedded in voice-activated routing systems at client decision centres, voice dealing on mobile phones, and plenty of alternative everyday applications. A strong speech-recognition system combines accuracy of identification with the flexibility to strain noise and adapt to alternative acoustic conditions, like the speaker's speech rate and accent. Coming up with a strong speech-recognition rule could be a complicated task requiring careful information of signal process and applied math modelling.

This article demonstrates a progress that uses inbuilt practicality in MATLAB® associated connected product to develop the rule for an isolated digit recognition system. The system is speaker-dependent — that is, it acknowledges speech solely from one specific speaker's voice.

INDIVIDUAL CONTRIBUTION

CODING: AMAN PANDEY(17BEC0406)
ARJUN T(17BEC0191)

DEBUGGING: AYUSH KUMAR AGARWAL(17BEC0668)

REPORT: VINEETH (17BEC0711)
SAI AAKASH BHATNAGAR(17BEC0593)

RESEARCH: PALAK KHATRI(17BEC0383)

VIDEO: VINEETH (17BEC0711)
SAI AAKASH BHATNAGAR(17BEC0593)
PALAK KHATRI(17BEC0383)

RECORDING: AYUSH KUMAR AGARWAL(17BEC0668)

CLASSIFYING SPEECH-RECOGNITION SYSTEMS

Most speech-recognition systems are classified as isolated or continuous. Isolated word recognition needs a short pause between every word, whereas continuous speech recognition doesn't. Speech-recognition systems may be additional

classified as speaker-dependent or speaker-independent. A speaker-dependent system solely acknowledges speech from one specific speaker's voice, whereas a speaker-independent system will acknowledge speech from anybody.

Development Workflow

Our development workflow consists of three steps:

- Speech acquisition
- Speech analysis

User interface development

Acquiring Speech

For coaching, speech is nonheritable from an electro-acoustic transducer and brought into the event setting for offline analysis. For testing, speech is ceaselessly streamed into the setting for on-line process.

During the coaching stage, it's necessary to record perennial utterances of every digit within the wordbook. for instance, we have a tendency to repeat the word ‘one’ persistently with an interruption between every vocalization.

Analysis of MATLAB Code

The main concept used in the following code is correlation

Correlation is normally used in signal processing. It is also known as the dot product of two signals. It can also be used in pattern recognition. The code revolves around the recognition of audio files present in the sample files folder. The wavread function is used instead of the auread function to load a wave flee specified by the string filename, returning the sample data in 'y'. The wavextension is appended if no extension is given. Amplitude values are in the range [-1, +1].

MATLAB Code

```
clear all clc;

x=1;

while x==1,

    x=isempty (input ('Press enter to start program or press zero to quit. '));

    ghighE=1318.1505;

    gB=987.7669;

    gG=783.9911;

    gD=587.3297;

    gA=440;

    glowE=329.6277;

    guitar=[ghighE,gB,gG,gD,gA,glowE];

    if x==1,

        p=input ('Which string are you adjusting? 1=e 2=B 3=G 4=D 5=A 6=E');

        w=guitar (p);

        y=isempty (input ('Press enter to record input signal or 0 to start over. '));

        % record input signal

        Fs=44100;

        T=0:1:Fs/2-1;

        RecordObject=audiorecorder(Fs,16,1);
        record(RecordObject,2);
        pause(3);
        %record(RecordObject,'off');
```

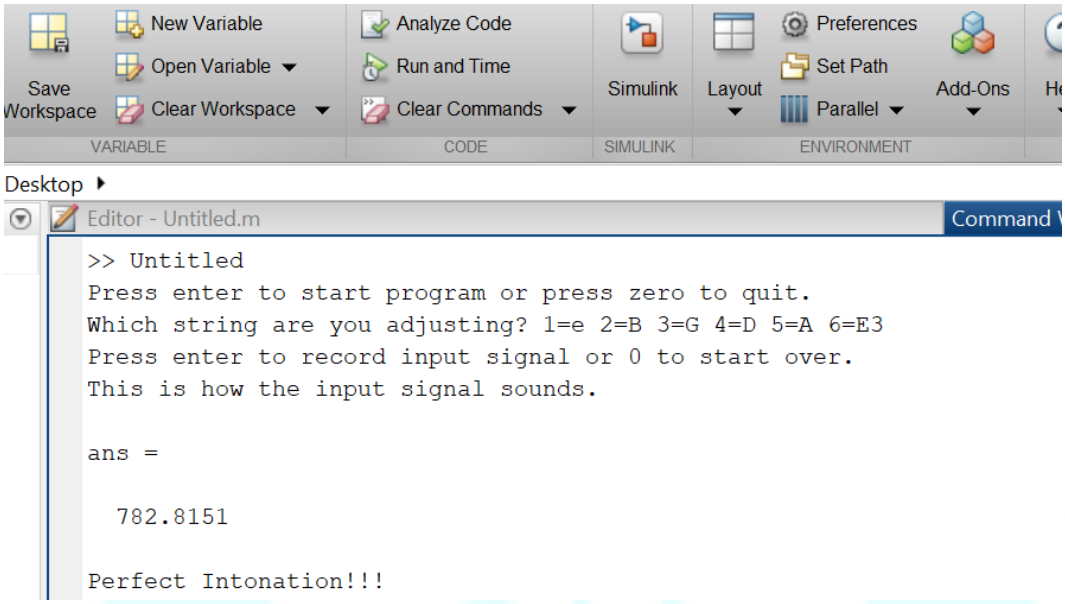
```
I=getaudiodata(RecordObject,'double');

% to compare frequencies of a stored save file
% let's hear the input signal
display ('This is how the input signal sounds. ');
sound (I,Fs);
% fft of input signal
J=fft (I)/size(I,1);
K=0:1:Fs/2-1;
while y==1,
   if p==6
      for i=400:size(J,1)
         J(i,1)=0;
      end
   else
      for i=1500:size(J,1)
         J(i,1)=0;
      end
   end
   %Plotting
   subplot(2,1,1);plot(I);
   subplot(2,1,2);plot(K,2*real(J(1:Fs/2)));
   if p==6
      axis([ 200 500 -0.01 0.01])
```

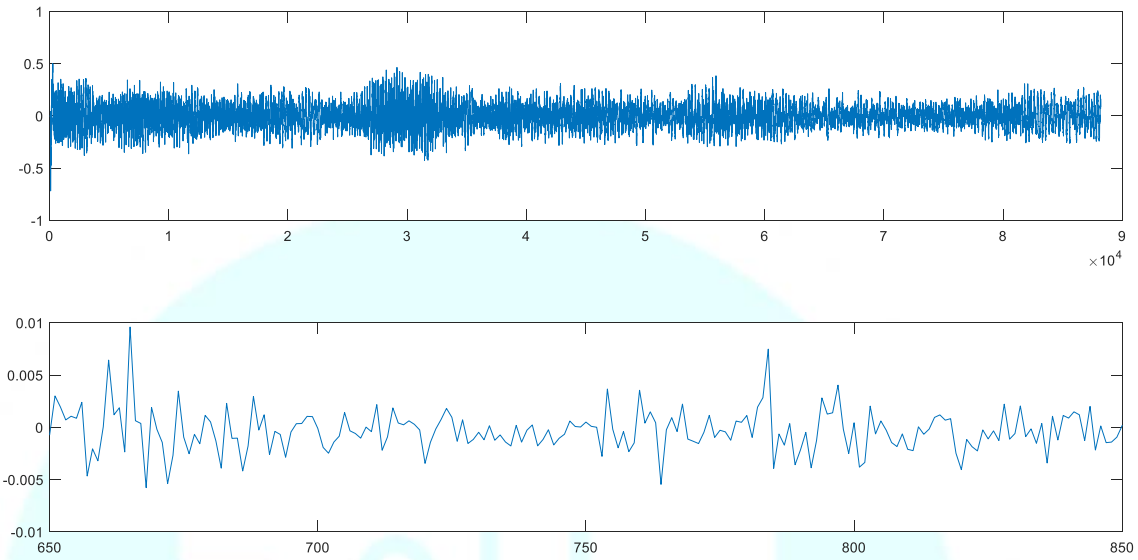
```
    for i=400:size(J,1)
        J(i,1)=0;
    end
elseif p==5
    axis([ 400 550 -0.01 0.01])
    for i=550:size(J,1)
        J(i,1)=0;
    end
elseif p==4
    axis([ 450 700 -0.01 0.01])
    for i=700:size(J,1)
        J(i,1)=0;
    end
elseif p==3
    axis([ 650 850 -0.01 0.01])
    for i=900:size(J,1)
        J(i,1)=0;
    end
elseif p==2
    axis([ 800 1100 -0.01 0.01])
    for i=1200:size(J,1)
        J(i,1)=0;
    end
elseif p==1
```

```
    axis([ 1200 1400 -0.01 0.01])

    for i=1500:size(J,1)

        J(i,1)=0;

    end

end

% xlabel('Frequency(Hz)'),ylabel('Amplitude(V)');

% title('input signal');
K=0:1:(Fs/2-1);

(1.0015*w);

K (J==max (J));

(0.9985*w)

z=((K (find (J==max (J)))-w)/w)*100;

if (1.0015*w)>K (find (J==max (J)))&& (0.9985*w) <K (find (J==max (J))),

    display ('Perfect Intonation!!!');

    disp (['The frequency of the input signal in Hz is, ']);
    disp((K (find (J==max (J)))))
    fprintf ('The correct frequency is % d  \n',w)
    fprintf ('Percent Error % d %% \n',z)

elseif w>K (find (J==max (J))),

    display ('Input frequency should be increased');

    disp (['The frequency of the input signal in Hz is, ']);
    disp((K (find (J==max (J)))))
    fprintf ('The correct frequency is % d  \n',w)
    fprintf ('Percent Error % d %% \n',z)

elseif w<K (J==max (J)),

    display ('Input frequency should be decreased');
```

```
            disp (['The frequency of the input signal in Hz is, ']);
            disp((K (find (J==max (J)))))
             fprintf ('The correct frequency is % d  \n',w)

            fprintf ('Percent Error % d %% \n',z)

        end

        y=isempty (input ('Make needed adjustments and press enter to record a new input signal or
0 to start over. '))

        if y==1,

            continue;

        else x=1;

        end

    end

  end

end
```

OUTPUT:


New Variable
Open Variable
Clear Workspace
Save Workspace
Analyze Code
Run and Time
Clear Commands
Simulink
Layout
Preferences
Set Path
Parallel
Add-Ons
VARIABLE
CODE
SIMULINK
ENVIRONMENT
Desktop
Editor - Untitled.m
Command
>> Untitled
Press enter to start program or press zero to quit.
Which string are you adjusting? 1=e 2=B 3=G 4=D 5=A 6=E3
Press enter to record input signal or 0 to start over.
This is how the input signal sounds.
ans =
782.8151
Perfect Intonation!!!



Graphical User Interface(GUI)

GUIs (also known as graphical user interfaces or UIs) provide point-and-click control of software applications, eliminating the need to learn a language or type commands in order to run the application.

MATLAB® apps are self-contained MATLAB programs with GUI front ends that automate a task or calculation. The GUI typically contains controls such as menus, toolbars, buttons, and sliders. Many MATLAB products, such as Curve Fitting Toolbox™, Signal Processing Toolbox™, and Control System Toolbox™ include apps with custom user interfaces. You can also create your own custom apps, including their corresponding UIs, for others to use.

Concepts Used in MATLAB Code

Correlation

Correlation is normally used in signal processing, where you need to compare two signals and need to find the similarity between them. It is also known as the dot product of those two signals. Correlation is a mathematical operation that is very similar to convolution. Just as with convolution, correlation

uses two signals to produce a third signal. This third signal is called the cross-correlation of the two input signals. If a signal is correlated with itself, the resulting signal is instead called the autocorrelation.

RESULT

The successful implementation of a guitar tuner inside of Matlab. Without the use of FFT this would not be possible. This is a more acurate tuner than those that can be purchased at a guitar shop. As it have more sensitive microphone which can detect frequency over a better range

References

Youtube : https://www.youtube.com/watch?v=Bb0L8o01mA0

Wikipediea : https://en.wikipedia.org/wiki/Electronic_tuner

Quora: https://www.quora.com/What-are-some-good-projects-for-an-undergrad-signals-and-systems-class

github