Onur Aydın
20902097
EEE 424 - 01

EEE 424 – Digital Signal Processing

Project Report

Project Description

Project Name: Note Recognition System

Description: "Note Recognition System" is a MATLAB based audio signal processing project that tries to extract the notes of an audio or music signal. The system is mostly based on record signals that include only one instrument. As a specific instrument, piano is chosen for this project, but it works well with other instruments too like violin, flute, trumpet and so on. Main drawback of the project is recognizing multiple notes that are played simultaneously. Even though, project's performance is very high with singular notes, problem becomes more complex and performance decreases as increasing the number of notes played at once. Therefore, framing is used only for singular notes. For chord or multiple notes recognition, only one frame is used. Therefore audio files have limited duration for multiple note recognition system. More sophisticated techniques can be used in order to increase the performance and recognize multiple notes in long duration.

Audio Features

Throughout this project, used test audio files are in the format of "Waveform Audio File Format" (WAV). Most of the records have 2-channel of 44100 kHz sampling rate and 16 bits per sample.

Music Theory

It is a good point to introduce the "harmonics" and "Fundamental Frequency" terms. Harmonics is the result of the resonance vibration at one of the natural frequencies. Harmonics shows the characteristics of a sound like its note, instrument and so on. At any frequency other than a harmonic frequency, the resulting disturbance of the medium is irregular and non-repeating. Fundamental frequency is roughly the harmonic that has lowest frequency, or in other words longest wavelength, value of a wave or signal. Fundamental frequency is also called first harmonic. [1] Each note has a unique fundamental frequency; they are listed in the appendix part; and it increases in a logarithmic trend as frequency increases. Here is the formula for finding n^{th} key's fundamental frequency key. 440 Hz is the frequency value of 49^{th} key A4, the middle key.

$$f(n) = \left(\sqrt[12]{2}\right)^{n-49} \times 440\,\mathrm{Hz}$$

Harmonics of any note have frequencies at integer multiple of fundamental frequency. For instance, C3 note has fundamental frequency of 130 Hz and harmonics of 130 Hz, 260 Hz, 520 Hz, 1040Hz and so on and according to characteristic of the instrument, weights of the harmonics may vary. In addition, C4 note has fundamental frequency of 260 Hz and harmonics of 260 Hz 520 Hz and so on. Therefore, in some situations, it is hard to detect whether only C3 is played or C3 and C4 is played

simultaneously. Therefore, it is a good way to look weights of the harmonics and compare and decide them according to weights.

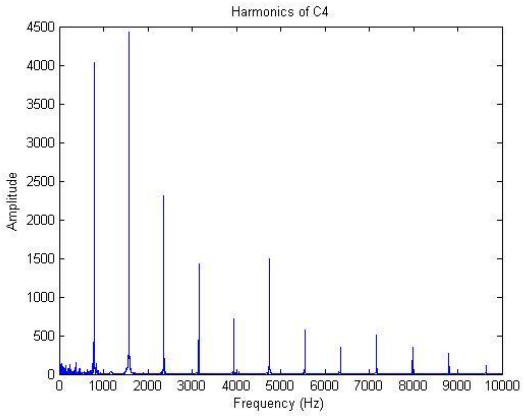

Figure 1 – Harmonics of C4 note

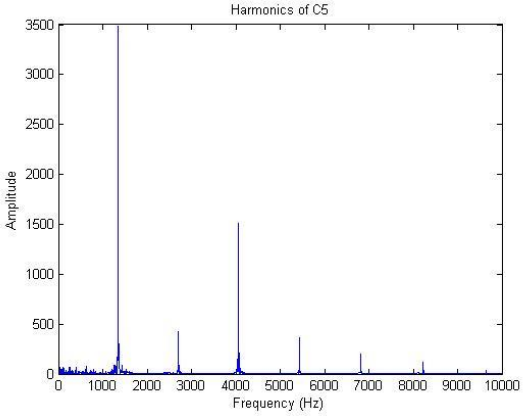

Figure 2 – Harmonics of C5 note

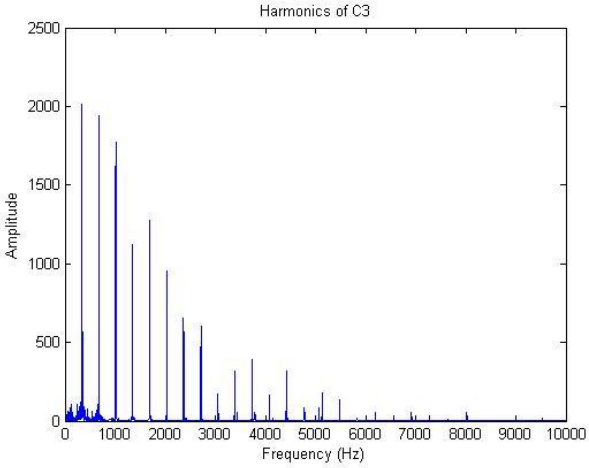

Figure 3 – Harmonics of C3 note

Moreover, another important point for this part is to understand the difference between playing only one note and multiple notes. When notes are played one by one, extracted frequency content is observed clearly. On the other hand, multiple notes played simultaneously bring extra harmonics and frequency content becomes more complicated.

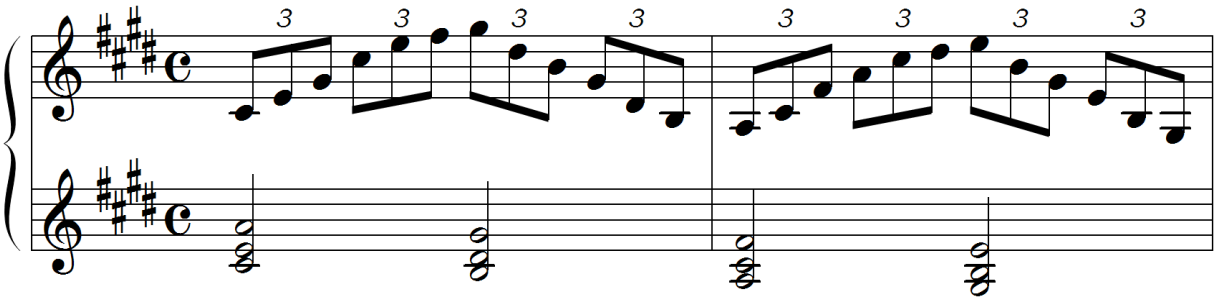

Figure 4 – Claude Debussy's *Premiere Arabesque* – Example of playing notes one-by-one

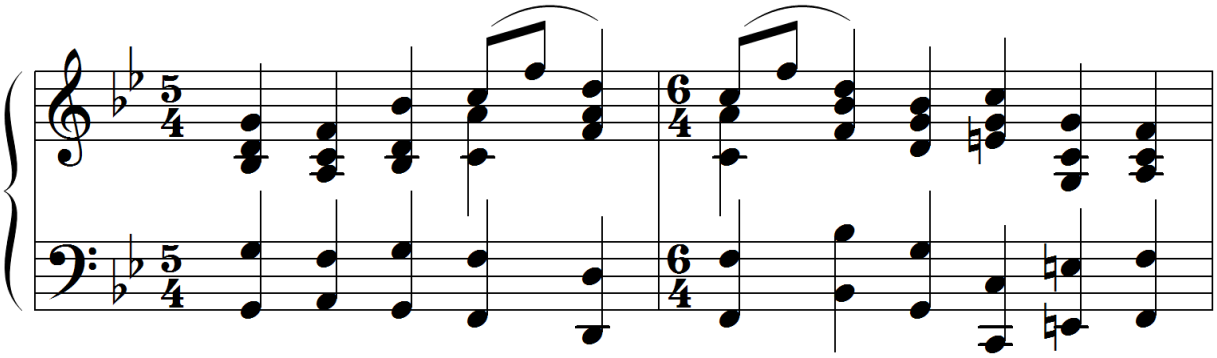

Figure 5 – Mussorgsky's *Pictures at an Exhibition* – Example of playing multiple notes

Design Decision

Sampling

In general, sampling is defined as taking measurements of a signal in certain time intervals. Sampling frequency or sampling rate is the number of samples obtained in 1 second. Mostly, sampling is applied to analog signals like speech, audio. In analog signals, there are infinite numbers of value that represent the signal completely but, it is not possible to store infinite number of values in computer systems. However, according to Shannon's theorem, it is possible to represent complete signal with just a few samples by sampling theorem.

In digital audio processing, most used sampling rates are 44.1 KHz, 22.05 kHz, 48 kHz, 88.2 kHz, and so on. However, most of audio encoding formats and audio CDs use 44.1 kHz. Most important reason of this value is frequency range of human ears can hear is 20Hz to 20 kHz and according to Nyquist-Shannon sampling theorem, sampling frequency must be at least two times bandwidth of the signal. Therefore, sampling frequency must be at least 40 kHz and 44.1 kHz is well enough.[2]

Audio Channel

Most of used audio files use 2 channels. In order to process the signals without any information loss, two channels are summed up and multiplied with some integer constant c because of strength the signal, but it is not necessary at all. Moreover, rather than summing up two channels, also their mean may be used.

Framing

Framing is dividing the audio signals into small portions and then processing each portion separately. It is important to take care of choosing frame length in order to get stationary frames. By framing, long audio signals are enabled to analyze. However, in this project, framing is used only for singular note recognition. For multiple notes, frames interfere with each other too much and results are affected terribly. Therefore, for multiple note recognition, only one frame is used and more complex framing methods must be developed. Moreover, in other perspective, framing is a type of windowing with rectangular window function which is not realizable in real life because of its sinc component in frequency domain.

Windowing

Windowing is multiplying the signal with a windowing function in time domain. The main objective of the windowing is obtaining better frequency content after taking DFT. By windowing, side lobes and leakage in frequency domain decrease and main frequency content emerges. There are different types of windowing functions such as Hamming, Hanning, rectangular, Kaiser and so on but for this application, Hamming window is well enough, because; it suppresses side lobes better.[3]

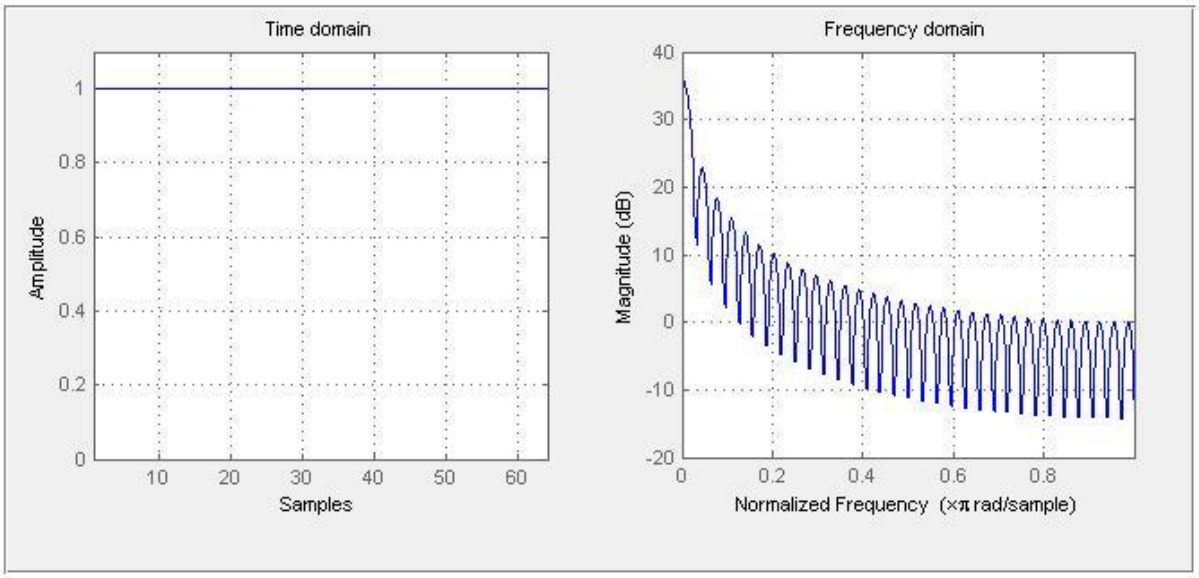

Figure 6 – Rectangular Window

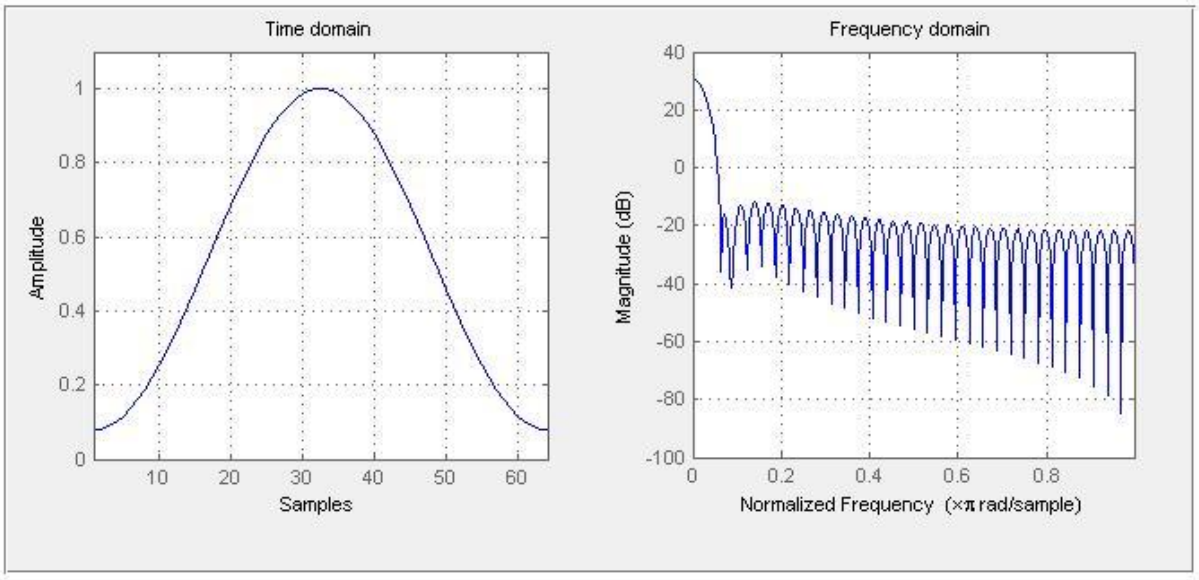

Figure 7 – Hamming Window

Discrete Fourier Transform

Audio signals keep their characteristics features inside the Fourier domain. By looking frequency domain, it is possible to differentiate signals from each other. Discrete Fourier Transform is a tool to transform the signals onto the frequency domain. Its implementation which has complexity of $o(n^2)$ is too costly. However, there are better algorithms for calculating Discrete Fourier Transform which are called as Fast Fourier Transform (FFT). Its complexity is o(nlog(n)) which is much better than classical implementation. FFT gives us a frequency content from 0 to 2Fs, where Fs is sampling frequency and it is symmetric about Fs. Therefore, just half of the FFT result is adequate. Also, just 0-5kHz interval is important for our application. Hence, we can eliminate others.[4] [5]

Frequency Band

After taking FFT of the signal, it is needed to determine the harmonics of the audio. In order to do this perfectly, according to fundamental frequency of notes, frequency scale is divided into the bands. Band limits are put on the middle of two notes' frequency values. Then, in order to get better band-frequency content, each band's mean value is multiplied with itself. By this way, each band's content emerged better. There are also other ways to increase the quality of band contents like summation, summation and taking power of 2,3 and so on, but taking power of expected values inside the band is the best way. In other words, in general perspective, frequency content is divided into 88 bands which is the number of key on a classical grand piano and for each coming audio, fundamental frequency and just a few harmonics content can be observed.

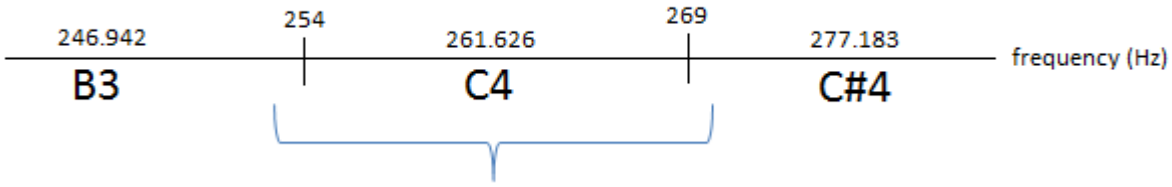

Figure x – Representation of frequency band for C4 note

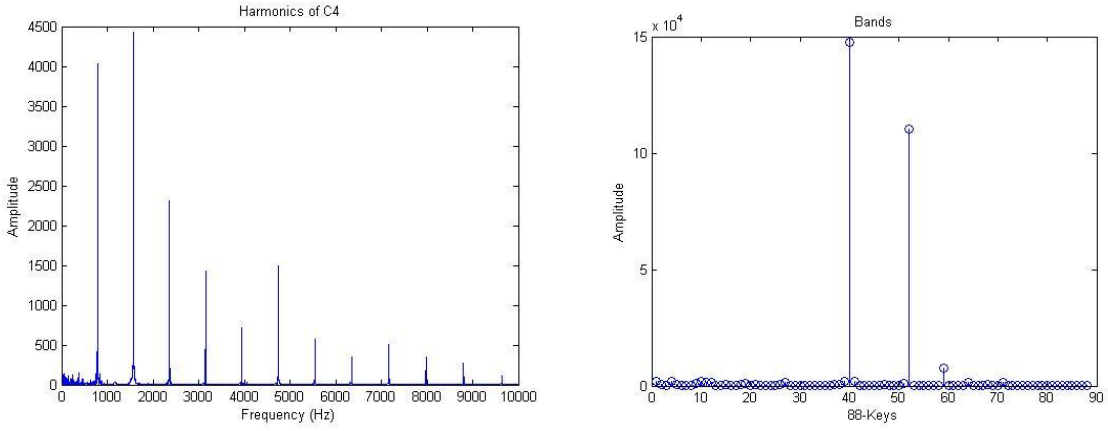

Figure 8 – Comparison between frequency content and band content for C4 note

From figure x, it can be observed that while there are lots of harmonics coming from frequency content, number of harmonics decrease to two in band content because of mathematical scaling. By this way, content may become clear and decision performance may increase.

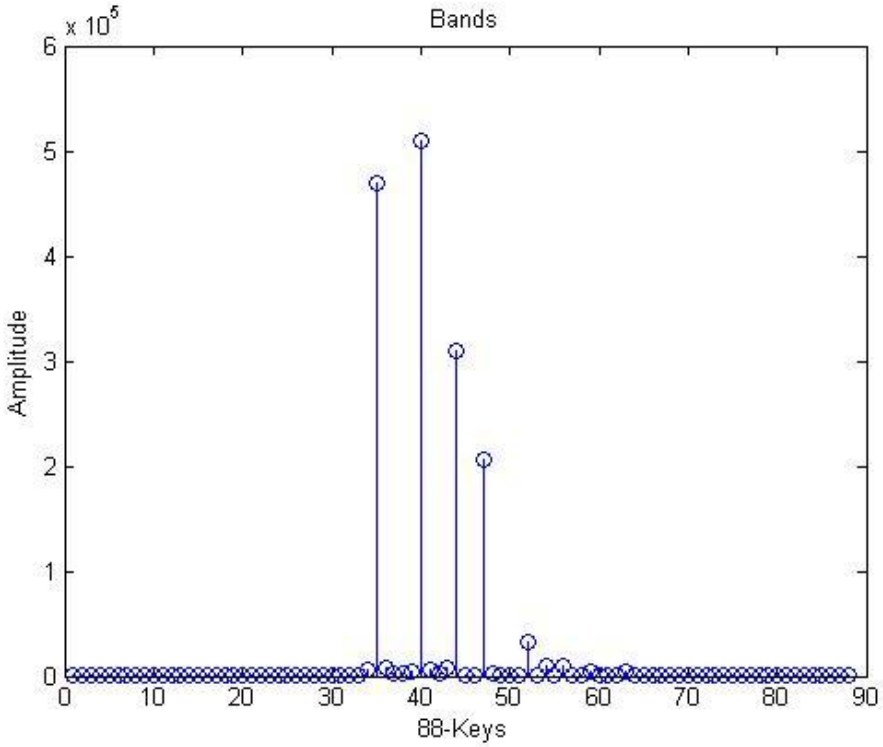

Figure 9 – An example of band content of a chord

By observing the band content in figure x, we can say that pushed keys are: 35th, 40th, 44th, 47th keys. These keys are: G3, C4, E4 and G4 respectively. W can say that directly lower notes; G3, C4 and E4; are played. However, for G4, we need to decide whether it is pushed or it is just a harmonic of G3.

Decision

There are different kinds of decision techniques that can be applied in my project, but I developed two decision techniques. First decision technique is taking the harmonic that has maximum amplitude. This technique is useful for if only one note is played, otherwise it doesn't work. Therefore, we need to know the content of the audio file before applying it. This is the most basic and very limited technique, but very efficient one. One simple check for this technique is that for example, max harmonic is C4 note value with 10A and if there is another harmonic let say C3 note value with 9A, we can say that max harmonic C4 is second harmonic of C3, therefore, pushed key is C3. Otherwise, C4 decision is kept. By this way, method's success increases a bit more.

Second method is more general than previous one. In this method, a threshold value is put to the band content and harmonics that have amplitude bigger than this threshold value are taken. Among this harmonics, decision is made. This method is not only for records that are played one by one, but

also for multiple notes. This one is more sophisticated and general technique, but for some files, efficiency is lower than first technique. Also, like first situation, simple check for harmonics can be done.

Test Results

Note Recognition

Test 1

File Name: Records/Chrome.wav

Description: From C8 to A0, all notes are played in an order.

Frame Length: 4410*2

Test notes: From C8 to C3, it can recognize the all notes nearly perfectly except a few of them cannot be processed because of their low amplitude. From C3, some errors start especially octave errors. For lowest notes, harmonics become complex and number wrong decisions reaches to its peak value. Considerable amount of notes are recognized, there are some problems occur on lower notes.

General result: Successful

Test 2

File Name: Records/Singular.wav

Description: From C4 to C6, all fundamental notes (C, D, E, F, G, A, B) are played in an order.

Frame Length: 4410*2

Test notes: Nearly all notes are recognized successfully. Just a few reflections of notes on frames give error, but it is not really vital.

General result: Successful

Test 3

File Name: Records/major.wav

Description: From C4 to C5, all fundamental notes (C, D, E, F, G, A, B) are played in an order.

Frame Length: 4410*2

Test notes: In this record, a different instrument is used rather than piano. This instrument sounds like a flute which has simpler harmonic content rather than piano. Therefore, it is highly expected to get perfect result. As a result, all notes are recognized perfectly.

General result: Successful

Test 4

File Name: Records/Plug in baby.wav

Description: A melody from Muse-Plug in baby song is played in piano from two different octaves.

Frame Length: 4410*2

Test notes: Even if some important notes couldn't be detected, success rate is considerably high. Success rate decreases, especially in accelerating sections. It is directly related to frame length. It is more complex record than others; therefore, it is understandable to get lower success rates. Also note that second part is played in higher octave and it gives better results because of higher notes have longer bands and clear harmonics.

General result: Successful-

Chord Recognition

Test 5

File Name: Records/Chord2.wav

Description: A chord F3-D4 is played

Test notes: In first phase, three harmonics F3, D4 and F4 are found and in second phase F4 is recognized is as a harmonic of F3. Therefore, it is eliminated and correct result is obtained.

General result: Successful

Test 6

File Name: Records/Chord3.wav

Description: A chord E3-F3-B3 is played

Test notes: In harmonics E3-F3-B3-E4-F4 are recognized but they didn't eliminated. However, it was very noise and complex record and just harmonics didn't eliminated.

General result: Successful--

Test 7

File Name: Records/Chord4.wav

Description: A chord G3-C4-E4 is played

Test notes: In harmonics G3-C4-E4-G4 are recognized and final harmonic G4 of G3 is eliminated successfully.

General result: Successful

Test 8

File Name: Records/Es.wav

Description: Two E notes are played simultaneously.

Test notes: In harmonics, there are E3 and E4 and it is recognized that E4 is not a harmonics of E3; therefore, it gives E3-E4 result.

General result: Successful

Conclusion

This project was helpful for me to use and understand fully the topics in digital signal processing course. In addition, I introduced with digital audio processing. Even if developed product is not so sophisticated, in limited time I tried to understand and observe the limits, difficulties and feasibility. As conclusion, I developed a product that fully recognize singular notes frame by frame and recognize multiple notes just one frame. By using more sophisticated methods, multiple note recognition can be developed. Finally, here are some useful ideas to improve the performance of the project:

- Initially, by knowing the scale of recorded song or melody, the bands can be regulated optimally, but it is a drawback for usability.
- In time domain, framing can be controlled in order to decrease the reflections of notes for next frame.
- According to pace of music, frame length can be changed in order to increase the performance, but is also a drawback for usability.
- User can be able to correct errors after implementation.
- Each push can be standardized by an algorithm. Therefore, loss of notes can be prevented.
- Directly using MIDI channel is a straightforward way and easier way for digital instruments. However, for analog instruments, this type of project is a necessity.

Appendices

Appendix A - Pinao Keys and Frequency Values

Key	Note	Frequency (Hz)	Bandwidth (Hz)		Key	Note	Frequency (Hz)	Bandwidth (Hz)
1	A0	27.500	20 - 28		49	A 4	440.000	428 – 453
2	A#0	29.135	28 - 30		50	A#4	466.164	453 - 480
3	B0	30.868	30 - 32		51	B 4	493.883	480 - 509
4	C 1	32.703	32 - 34		52	C 5	523.251	509 - 539
5	C#1	34.648	34 - 36		53	C#5	554.365	539 - 571
6	D 1	36.708	36 - 38		54	D 5	587.330	571 - 605
7	D#1	38.891	38 - 40		55	D#5	622.254	605 - 641
8	E 1	41.203	40 - 42		56	E 5	659.255	641 - 679
9	F 1	43.654	42 - 45		57	F 5	698.456	679 - 719
10	F#1	46.249	45 - 48		58	F#5	739.989	719 - 762
11	G 1	48.999	48 - 50		59	G 5	783.991	762 - 807
12	G#1	51.913	50 - 53		60	G#5	830.609	807 - 855
13	A 1	55.000	53 - 57		61	A 5	880.000	855 - 906
14	A#1	58.270	57 - 60		62	A#5	932.328	906 - 960
15	B 1	61.735	60 - 64		63	B 5	987.767	960 - 1017
16	C 2	65.406	64 - 67		64	C 6	1046.502	1017 - 1078
17	C#2	69.296	67 - 71		65	C#6	1108.731	1078 - 1142
18	D 2	73.416	71 - 76		66	D 6	1174.659	1142 - 1210
19	D#2	77.782	76 - 80		67	D#6	1244.508	1210 - 1282
20	E 2	82.407	80 - 85		68	E 6	1318.510	1282 - 1358
21	F 2	87.307	85 - 90		69	F 6	1396.913	1358 - 1438
22	F#2	92.499	90 - 95		70	F#6	1479.978	1438 - 1524
23	G 2	97.999	95 - 101		71	G 6	1567.982	1524 - 1615
24	G#2	103.826	101 - 107		72	G#6	1661.219	1615 - 1711
25	A 2	110.000	107 - 113		73	A 6	1760.000	1711 - 1812
26	A#2	116.541	113 - 120		74	A#6	1864.655	1812 - 1920
27	B 2	123.471	120 - 127		75	B 6	1975.533	1920 - 2034
28	C 3	130.813	127 - 135		76	C 7	2093.005	2034 - 2155
29	C#3	138.591	135 - 143		77	C#7	2217.461	2155 - 2283
30	D 3	146.832	143 - 151		78	D 7	2349.318	2283 - 2419
31	D#3	155.563	151 - 160		79	D#7	2489.016	2419 - 2563
32	E 3	164.814	160 - 170		80	E 7	2637.020	2563 - 2715
33	F 3	174.614	170 - 180		81	F 7	2793.826	2715 - 2877
34	F#3	184.997	180 - 190		82	F#7	2959.955	2877 - 3048
35	G 3	195.998	190 - 202		83	G 7	3135.963	3048 - 3229
36	G#3	207.652	202 - 214		84	G#7	3322.438	3229 - 3421
37	A 3	220.000	214 - 227		85	A 7	3520.000	3421 - 3625
38	A#3	233.082	227 - 240		86	A#7	3729.310	3625 - 3840
39	B 3	246.942	240 - 254		87	B 7	3951.066	3840 - 4069
40	C 4	261.626	254 - 269		88	C 8	4186.009	4069 - 4500
41	C#4	277.183	269 - 285					
42	D 4	293.665	285 - 302					
43	D#4	311.127	302 - 320					
44	E 4	329.628	320 - 339					
45	F 4	349.228	339 - 360					
46	F#4	369.994	360 - 381					
47	G 4	391.995	381 - 404					
48	G#4	415.305	404 – 428					

Appendix B - Matlab Codes

Note Recognition

```
clear; clc; close all;
% Note Recognition
%% Note Initialization
mainNames = char('C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B');
names = char('A0', 'A#0' ,'B0');
A0 = 27.5;
ind = 4;
for i = 0:87
    data(i+1) = A0 * (2^(1/12))^i;
    if i>2
         a = [mainNames( rem((ind - 4), 12)+1,:)  num2str(fix((ind - 4)/12)+1)];
         names = char(names, a);
         ind = ind + 1;
    end
end

% Band Initialization
bands(1) = 20;
for i = 1:87
    bands(i+1) = +7+(data(i) + data(i+1))/2;
end
bands(89) = 4500;

%%
fileName = 'Records/Plug in baby.wav';  % File name
[y, Fs] = wavread(fileName);            % Read audio file
y = (y(:,1) + y(:,2))*4;                % Decrease 2 channels to 1
%y = (y(:,1));
y(1:2:end) = 0;                         % Do decimation

frameLength = 4410*2; % 2 Güzel oldu

endPart = frameLength*ceil(length(y)/frameLength);  % complete the last frame
y(length(y)+1 : endPart) = 0;

f = linspace(1,Fs,frameLength);

%%
harmonics = 0;
for i = 1:round(length(y)/frameLength)      % For each frame
    % Divide audio into frames
    frames(i, 1:frameLength) = y( (frameLength*(i-1)+1):(frameLength*i) )';

    frame = y( (frameLength*(i-1)+1):(frameLength*i) )';
    frame = frame .* Hamming(length(frame))';    % Hamming Window
    fframe = abs(fft(frame));                    % FFT

    fp = sum(fframe);
    p = sum(abs(frame));
    b = true;
    if(p < 200 || fp < 1000) % Put a threshold for processing
       b = false;
    end

    % Bands
    for i=1:88
        freqBand(i) = mean(fframe( round(bands(i)/(Fs/frameLength)
):round(bands(i+1)/(Fs/frameLength))))^2;
    end
```

```
    % Plotting
    subplot(3,1,1)
    stem(freqBand)
    subplot(3,1,2)
    plot(fframe)
    xlim([0,500])
    subplot(3,1,3)
    plot(frame)
    ylim([-1 1])

    hold off
    pause(0.1)
    wavplay(frame,Fs)

    % Desicion
    m = find(freqBand == max(freqBand(:)));
    if(b) disp(names(m,:));     % Print the result
    else disp('.'); end

    if(b)
        index = 1;
        for i = 1:88
            if(freqBand(i) > 2000)
                harmonics(index) = i;
                index = index+1;
            end
        end
    end
end
```

Chord Recognition

```
Clear; clc; close all;
% Chord Recognition
%% Note Initialization
mainNames = char('C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B');
names = char('A0', 'A#0' ,'B0');
A0 = 27.5;
ind = 4;
for i = 0:87
    data(i+1) = A0 * (2^(1/12))^i;
    if i>2
         a = [mainNames( rem((ind - 4), 12)+1,:)  num2str(fix((ind - 4)/12)+1)];
         names = char(names, a);
         ind = ind + 1;
    end
end

% Band Initialization
bands(1) = 20;
for i = 1:87
    bands(i+1) = +0+(data(i) + data(i+1))/2;
end
bands(89) = 4500;

%%
fileName = 'Records/Es.wav';     % File name
[y, Fs] = wavread(fileName);     % Read audio file
wavplay(y,Fs)                    % Play audio file
y = (y(:,1) + y(:,2))*2;         % Decrease 2 channels to 1

fy = abs(fft(y.*Hamming(length(y))));    % Hamming + FFT
```

```
frameLength = length(y);
for i=1:88                          % Create Bands
    freqBand(i) = mean(fy( round(bands(i)/(Fs/frameLength)
):round(bands(i+1)/(Fs/frameLength))))^2;
end

stem(freqBand)
xlabel('88-Keys'); ylabel('Amplitude'); title('Bands');

f = linspace(1,Fs,length(fy));

m = find(freqBand == max(freqBand(:)));
index = 1;
for i = 1:88                        % Find Harmonics
    if(freqBand(i) > 2000)
        harmonics(index) = i;
        index = index+1;
    end
end

rate = freqBand(m) ./ freqBand(harmonics);  % Put a rate
index = 1;                                  % According to rate
for i = 1:length(harmonics)                 % determine the harmonics
    if(rate(i) < 5)
        newHarmonics(index) = harmonics(i);
        index = index +1;
    end
end
harmonics = newHarmonics;
fharmonics = freqBand(harmonics);

disp(names(harmonics,:));

% Check octaves
index = 1;
remove = 0;
for i = 1:length(harmonics)
    for j = i:length(harmonics)
        if(i ~= j)
            a1 = names(harmonics(i),1:2);
            a2 = names(harmonics(j),1:2);

            if(a1 == a2)
                if(fharmonics(j)> fharmonics(i))
                    % Do nothing
                else
                    remove(index) = j;
                    index = index +1;
                end
            end
        end
    end
end
if(remove(1) ~= 0)
    harmonics(remove) = [];     % Remove if it is a harmonic of a note
end

disp('*****')               % Print
for i = 1:length(harmonics)
    if(harmonics(i) > 0)
        disp(names(harmonics(i),:));
    end
end
```

References

[1] physicsclassroom.com "Resonance and Standing Waves" [Online]. Available: http://www.physicsclassroom.com/class/sound/u11l4d.cfm. [Accessed: Jan. 11, 2014].

[2] Wikipedia “Sampling Rate” [Online]. Available: http://en.wikipedia.org/wiki/Sampling_rate [Accessed: Jan. 11, 2014].

[3] Wikipedi “Window Function” [Online]. Available: http://en.wikipedia.org/wiki/Window_function [Accessed: Jan. 11, 2014].

[4] Wikipedia “Discrete Fourier Transform” [Online]. Available: http://en.wikipedia.org/wiki/Discrete_Fourier_transform [Accessed: Jan. 11, 2014].

[5] Wikipedia “Fast Fourier Transform” [Online]. Available: http://en.wikipedia.org/wiki/Fast_Fourier_transform [Accessed: Jan. 11, 2014].