**The Chinese University of Hong Kong, Shenzhen**

# EIE 3510 Digital Signal Processing

# Term Project Report

**Li Jiayuan 117010120**
**Xiao Nan 117010303**

# Final Report

Li Jiayuan 117010120  Xiao Nan 117010303

## 1. Introduction

In this project, we developed a note detection system which can receive a piece of monophonic music played by piano and identify which note is present at each moment in the music. We utilized the frequency relationship between pitch and harmonics components in music and estimated the music notes by calculating the signal period within a short time window. Altogether 5 methods from time domain, frequency domain, and time-frequency domain are presented with detection accuracy all above 32/33 and detection error rate around lower than 22%. We also tried to apply down-sampling to reduce computational complexity and addressed our newly-found octave jump problem using subtraction method. Further improvements such as end-point detection and polyphonic music note detection are discussed as well.

## 2. Problem

Music note detection has a long history in music processing and Music Information Retrieval (MIR). It is about identifying the pitch component from the combination of different waves. As a basic and important perceptual feature of sound, pitch contains much information and it is important for us to separate it from music. Note detection is also important in speech and audio processing. By extracting this feature, it can facilitate other applications in speech processing such as speech recognition and speaker identification.

Before we talk about the details of detection algorithms, we first need to under several basic definitions about music signals and music notes.

**1) Music Signal:**

Music signal $S$ is the linear superposition of waves with different frequencies. The one with the lowest frequency is called **pitch**, which decides the sound frequency human perceive. The corresponding frequency $f_0$ is called fundamental frequency. The other waves are called **harmonics** and their frequencies are referred to $f_1, f_2 \ldots . f_n$. The pitch and harmonics altogether decide the timbre of sounds.

$$S = A_0 cos(2\pi f_0 t) + A_0 cos(2\pi f_0 t) + \cdots + A_0 cos(2\pi f_0 t)$$

The frequency relationship between pitch and harmonics components is essential to understand the note detection algorithm. In fact, the frequencies of harmonics are always the **integer multiples** of fundamental frequency $f_0$, which can be derived from the sound generating process as standing waves.

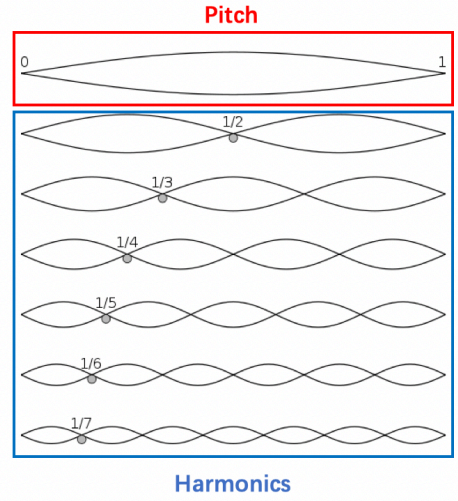



Figure 1 Pitch and harmonics components

When a sound wave is generated, it will bounce back and forth between walls. This kind of situation where the end points of waves are fixed is called **fixed-ends oscillation**. In between two fixed point, we can have various oscillations as shown in Fig. 1, where the red-boxed wave with the largest wave length corresponds to the fundamental frequency $f_0$ and the others corresponds to harmonic waves. It's easy to derive from the figure that, the wavelength relationship between pitch and harmonics is:

$$\lambda_n = \frac{1}{n+1}\lambda_0$$

whereas frequency and wavelength are related by the equation $f = \frac{c}{\lambda}$, where $c$ is the speed of sound – approximately a constant in constant temperature. Therefore, we can derive the frequency relationship between pitch and harmonics as:

$$f_n = \frac{c}{\lambda_n} = \frac{f_0 * \lambda_0}{\lambda_n} = (n+1)f_0$$

As a result, the final composition wave will have the same frequency with the lowest frequency, which is the fundamental frequency $f_0$. In general, we will utilize this derived property to perform piano note detection.

**2) Music Note:**

Music Notes are theoretical symbols such as C3 and C4 that are related to distinct frequencies. In piano, there are altogether 88 keys with frequencies ranging from 27.5Hz (A0) to 4186Hz (C8).

| Key number | Helmholtz name | Scientific name | Frequency (Hz) |
|---|---|---|---|
| 88 | c′′′′′ 5-line octave | C8 Eighth octave | 4186.01 |
| 87 | b′′′′ | B7 | 3951.07 |
| 86 | a♯′′′′/b♭′′′′ | A♯7/B♭7 | 3729.31 |
| 85 | a′′′′ | A7 | 3520 |
| 84 | g♯′′′′/a♭′′′′ | G♯7/A♭7 | 3322.44 |
| 83 | g′′′′ | G7 | 3135.96 |
| 82 | f♯′′′′/g♭′′′′ | F♯7/G♭7 | 2959.96 |
| 81 | f′′′′ | F7 | 2793.83 |
| 80 | e′′′′ | E7 | 2637.02 |
| 79 | d♯′′′′/e♭′′′′ | D♯7/E♭7 | 2489.02 |
| 78 | d′′′′ | D7 | 2349.32 |
| 77 | c♯′′′′/d♭′′′′ | C♯7/D♭7 | 2217.46 |
| 76 | c′′′′ 4-line octave | C7 Double high C | 2093 |
| 75 | b′′′ | B6 | 1975.53 |
| 74 | a♯′′′/b♭′′′ | A♯6/B♭6 | 1864.66 |
| 73 | a′′′ | A6 | 1760 |
| 72 | g♯′′′/a♭′′′ | G♯6/A♭6 | 1661.22 |

Figure 2 Music Notes and Corresponding Frequencies

It's noted that between every octave (like A4 and A5), the note frequency will double. The frequency relationship between keys follow the **12-tone equal temperament**: an octave is equally divided into 12 semitones. Therefore, the semitone relationship between any note and the standard A4 can be formulated as:

$$\mathrm{N} = 12 * \log_2\left(\frac{f}{440}\right)$$

where $f$ is the frequency of any note.

In a word, our project is to detect **theoretical music note** (in the form of semitone relationship $N$) occurred in each time interval from piano **music signal** $\boldsymbol{S}$. More specifically, we will find the fundamental frequency $f_0$ of the sound wave and map it to the theoretical note, which is our detection result.

# 3. Approach

The structure of our project is shown below.

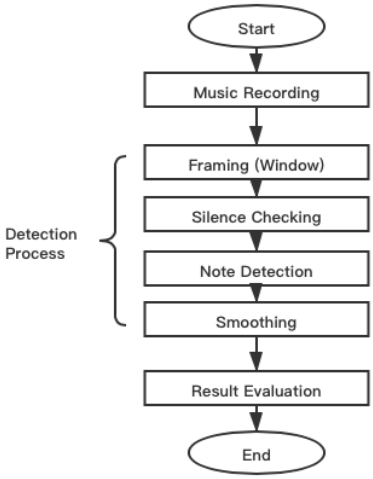



Figure 3 Project Structure

which includes 6 steps: music recording, framing, silence checking, note detection, smoothing, and result evaluation.

### 3.1. Music Recording

To facilitate later evaluation, we choose the Garage Band software to generate the music Fur Elise by Beethoven. We also choose the built-in Steinway grand piano to simulate the real piano sound. The midi interface and the corresponding music sheet are shown below:

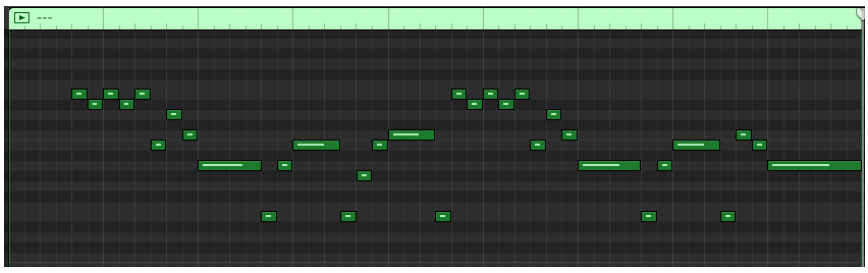


Figure 4 MIDI Interface for Our Music

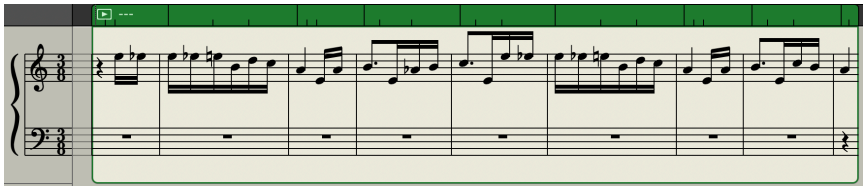


Figure 5 Music Sheet for Our Music

### 3.2. Framing

For framing, we take out a segment of music each time using window for analysis as there are different notes to detect at each time. There are two parameters of window to be decided: the window length and the window type.

The **window length** is typically taken to contain 10ms to 30ms of the signal. The basic principle is that in order to allow enough delay to perform auto-correlation, the window should contain at least three periods of signals. For piano, the lowest frequency and highest frequency are 27.5Hz and 4186Hz. Therefore, the window segment should contain three periods of the lowest frequency, which can be calculated to be 0.1091s. However, as in our project we didn't use all the piano notes, we may adjust the window size to accommodate three periods of our lowest frequency 329.628Hz. The smallest size of window is determined to be 1/329.628*3 = 0.0091s, which is around 402 samples with sampling rate being 44100Hz.

For the **window type**, we used Hamming window to do framing in order to reduce the "spectral leakage" as we learnt in the class.

### 3.3. Silence Checking

Before perform detection algorithm, we first need to check whether this frame is silent or not – whether there is sounding note in this frame. The basic principle is to calculate the energy in each frame.

$$\mathrm{E} = \sum_{i=0}^{n} |S(i)|^2$$

where n is the frame size.

If the energy is above a certain **threshold**, then we assume there is sounding note in this frame. The threshold should be determined from the amplitude of the music signal such that weak note should not be misidentified as noise. For a single case, it is also possible to try different thresholds based on the theoretical calculated one to achieve best performance.

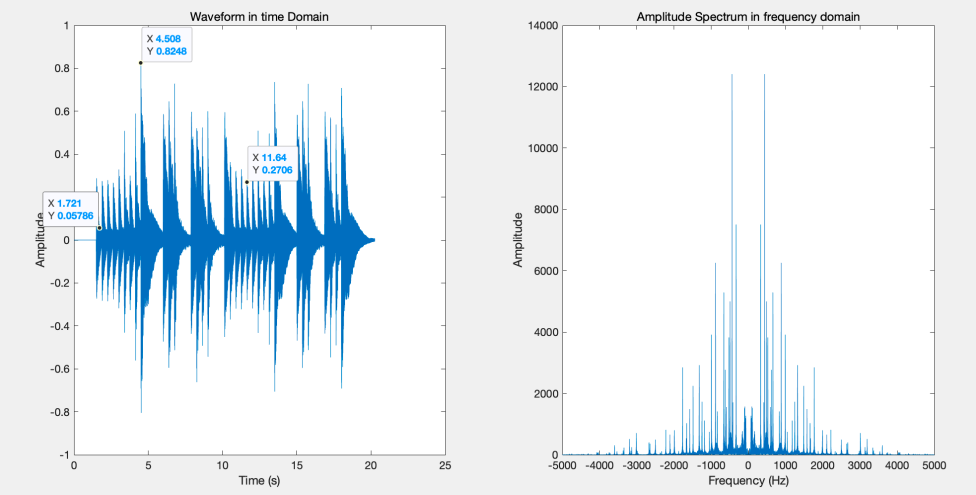



Figure 6 Signal Waveform in Time Domain and Freq. Domain

In our case, the amplitude of the smallest peak is 0.2706, which means that the maximum energy of the weakest sound is around 0.0732. Usually the signal ends around amplitude 0.058, whose energy is around 0.003. To be safe, we set the energy threshold to be 0.0001.

### 3.4. Note Detection

Note detection has various algorithms from the time domain, the frequency domain, and the time-frequency domain. Generally, they are based on the fact that the composite wave is periodic with period T.

#### 3.4.1 Time Domain Method 1 – ACF

This method is called the **Auto Correlation Function (ACF)**. The basic principle is to calculate the period T of the composite wave by computing the auto-correlation. Auto-correlation is essentially measuring the similarity between the original wave and its delayed version. The more similar these two waves, the higher the auto-correlation at that delay.

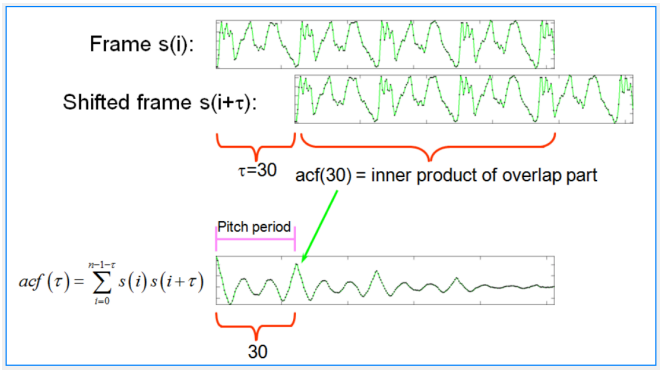



Figure 7 Auto Correlation Function

Theoretically, the auto-correlation result should have peaks at delay 0*T, 1*T, 2*T … n*T as the composite wave is periodic with a period of T. This means that we should select the first local maximum except for the peak in delay zero. We may define a **cut-off delay** such that the auto-correlation value will be equaled to the final value, so as to remove the peak at delay zero.

$$ACF(\tau) = ACF(\tau_0), \text{for } \tau < \tau_0$$

and the fundamental frequency $f_0$ can be determined by the equation $f_0 = 1 / \mathrm{T}$.

Nevertheless, it may be noticed that some notes may demonstrate a fake period of T/2. For example, we plot the waveform of the generated note C3 below. According to the theoretical frequency of C3 pitch, we can calculate the theoretical period to be T = 1/f = 1/130 = 0.0077. The period shown in the figure below can be calculated to be T=T1-T2= 0.008, which is very close to the theoretical result.

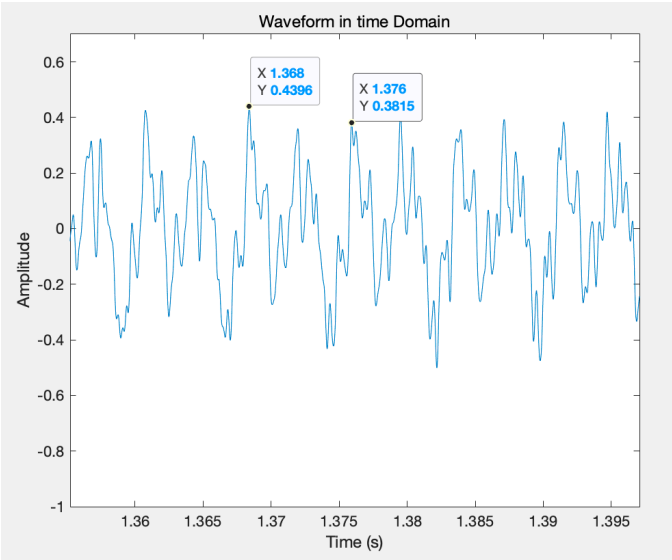



Figure 8 Detailed Waveform in Time Domain with Fake Period T/2

However, as shown above, the wave also seems to demonstrate another period with $T' = T/2$ as the waveform in half cycle is very similar. As a result, $ACF(\frac{T}{2})$ and ACF(T) may be very close, and the delay corresponds to the first local maximum will be T/2 and the detected frequency become the first harmonic $f_1 = 2f_0$.

In addition, ACF also suffer from the **tapering effect**, which means it is easily affected by the number of overlapping points. As a result, the auto-correlation in harmonic frequency $ACF(\tau_n)$ maybe higher than the one in the fundamental frequency $ACF(\tau)$ as harmonics has fewer delay and more overlapping points.

Therefore, to eliminate the influence of harmonics, we try to **normalize** the auto-correlation to flatten the harmonic peaks.

$$NACF(\tau) = \sum_{i=0}^{n-1-\tau} \frac{s(i)s(i+\tau)}{n-\tau}$$

According to the theory of auto-correlation, the normalized auto-correlation of harmonics should be **lower** than that of pitch, as the similarity between original wave and T-delayed wave is supposed to be zero and the similarity between original wave and T/n-delayed (left side of T) wave is supposed to be larger than zero. Moreover, considering that the amplitude of wave will decay as sound will fade away, the similarity between nT-delayed (right side of T) wave and the original wave will **decrease** with n, so as the normalized auto-correlation. As a result, we can simply detect the **maximum** of NACF instead of the first local maximum in the original ACF.

$$f = \frac{1}{\tau} = \frac{1}{index(\max(NACF(\tau)))}$$

### 3.4.2 Time Domain Method 2 – AMDF

The second method is called **Average Magnitude Difference Function (AMDF)**. It is essentially the same as the previous ACF that they both utilize the fact of the composite wave should be periodic with period T. Instead of calculating the similarity between original wave and delayed wave, AMDF calculates the magnitude difference between these two waves.

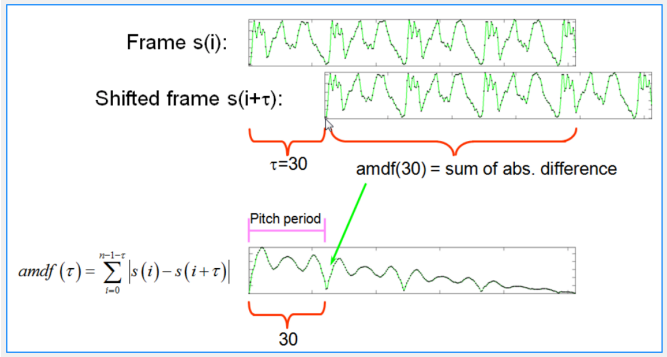



Figure 9 Average Magnitude Difference Function

Compared to ACF, AMDF is less sensitive to the amplitude of waves, as it is calculating the relative difference instead of absolute auto-correlation. The period detection is the same except for here we are looking for the first **local minimum** instead of local maximum except for the peak in delay zero. We also define a **cut-off delay** such that the auto-correlation value will be equaled to the final value, so as to remove the peak at delay zero.

$$AMDF(\tau) = AMDF(\tau_0), \text{for } \tau < \tau_0$$

and the fundamental frequency $f_0$ can be determined by the equation $f_0 = 1 / T$.

Similarly, in order to eliminate the tapering effect, we modified the original AMDF into normalized AMDF.

$$NAMDF(\tau) = \sum_{i=0}^{n-1-\tau} \frac{|s(i) - s(i+\tau)|}{n-\tau}$$

For real implementation, we just take the reciprocal of AMDF and detect the local maximum.

$$f = \frac{1}{\tau} = \frac{1}{index(\max\left(\frac{1}{NAMDF(\tau)}\right))}$$

### 3.4.3 Frequency Domain Method 1 – HPS

The first method in the frequency domain is called **Harmonic product spectrum (HPS)**. As we introduced before, the sound wave is composed of pitch and harmonics. Therefore, when we take the Fourier Transform of the signal, we should see a series of peaks which corresponds to the pitch and harmonics. Their spacing in the frequency domain is supposed be the fundamental frequency $f_0$ as the frequency of harmonic components are at integer multiples of the fundamental frequency.

$$\Delta f = f_1 - f_0 = f_2 - f_1 = \cdots = f_n - f_{n-1} = f_0$$

We can then measure the **coincidence** for harmonics and the index for the maximum is the frequency spacing, which is $f_0$.

$$f_0 = index(\max\left(Y(\omega)\right))$$

and the coincidence can be measured by **down-sampling** and the **multiplication** of each down-sampling spectrum, which gives the name of this method. The mathematical equation is given by:

$$Y(\omega) = \prod_{r=1}^{R} |X(\omega_r)|$$

where $X(\omega_r)$ is the rth down-sampling of the Fourier transform of the music segment, $Y(\omega)$ is the resulting periodic correlation array, and $R$ is the number of harmonics being considered.

For $R = 3$, the HPS procedure can be shown as the following:

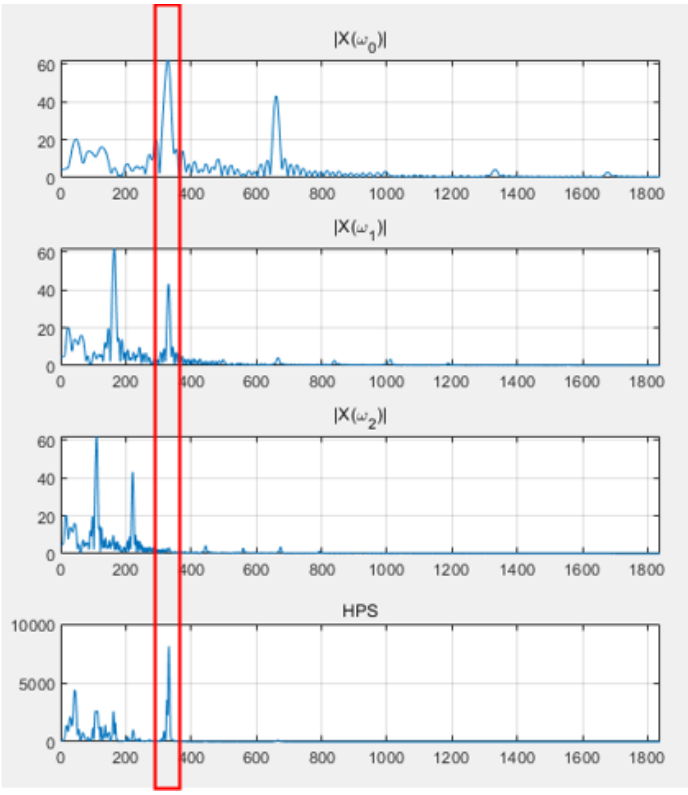



Figure 10 Frequency Spectrum for 0th, 1st ,2nd Down-sampling and Final HPS

We can see from the figure above that, after multiplying the spectrums the harmonic components in the right-hand side will be eliminated and the energy will be transferred to the fundamental frequency component, which is what we are detecting.

However, it is noticeable that both the **N (number of points in Fourier transform)** and **R (numbers of down-sampling)** may have huge impact on the results of coincidence. For N, as frequency resolution is given by Fs/N where Fs is the sampling frequency, so a larger N is required if we want to achieve a higher resolution. Also, as the number of points in each music segment is fix by the window size, here we use zero padding method to obtain a large N. For R, its choice depends on the property of the harmonics of the music segment. As the amplitude of harmonics will decrease with n, the nth harmonics maybe zero in the frequency domain. If we take n down-sampling, the spectrum of nth down-sampling will have zero amplitude inherited from the nth harmonics. Therefore, the multiplication result will become zero for the fundamental frequency $f_0$.

### 3.4.4 Frequency Domain Method 2 – Cepstrum

"Cepstrum" is a play on the word spectrum as it is simply a spectrum of a spectrum. The mathematical equation is defined as:

$$C(\tau) = F^{-1}\left(\log\left|F\left(x(t)\right)\right|^2\right)$$

where $C(\tau)$ is the Cepstrum of the music segment $x(t)$, $F^{-1}$ represent inverse Fourier transform and $F$ stands for Fourier transform.

By Wiener–Khinchin theorem, the Power Spectral Density $S(f) = |F(x(t))|^2$ has the following relation with autocorrelation function $ACF(\tau)$:

$$ACF(\tau) = F^{-1}(S(f)) = F^{-1}(|F(x(t))|^2)$$

We can see that $C(\tau)$ is similar to $ACF(\tau)$ except for a logarithmic scale conversion before the inverse Fourier transform, so similar to $ACF$, we can conclude that Cepstrum has property of measuring autocorrelation as well. Furthermore, since we focus on the periodic effects only and ignore the phase differences, to reduce computation complexity, we can simplify the equation of Cepstrum as:

$$C(\tau) = F^{-1}\left(\log\left|F\left(x(t)\right)\right|\right)$$

Then, similar to ACF, we can obtain the frequency $f$ by searching for local maximum and take the reciprocal.

$$f = \frac{1}{\tau} = \frac{1}{index(\max\left(C(\tau)\right))}$$

### 3.4.5 Time-frequency Domain Method – Wavelet Transform

In all the previous methods, we apply framing with **fix window size**, which means that we treat signal as stationary within the window. According to our analysis, the window size should be large enough to contain three periods of signals. However, if the note duration is very small, it is possible that our music segment will contain two or more notes which will definitely reduce the performance of our algorithms as they assume only one note is present in each frame.

Therefore, we encounter the **resolution dilemma** in time and frequency. This can also be explained by the Heisenberg Uncertainty Principle that, it's impossible to locate time and frequency precisely at the same time. As we can see from the following figures, **small window** will result in low frequency resolution (overlapping

in frequency domain) while **large window** will have low time resolution (overlapping in time domain) at low frequencies.

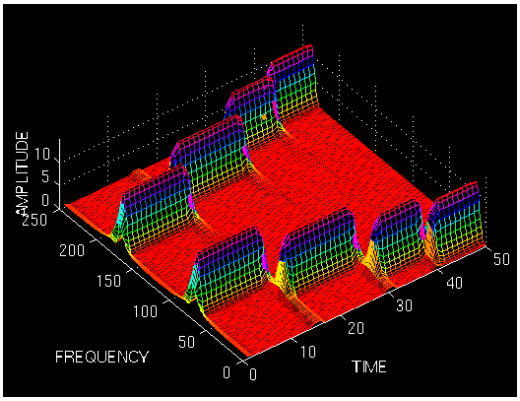


Figure 11 Result when applying Small Window

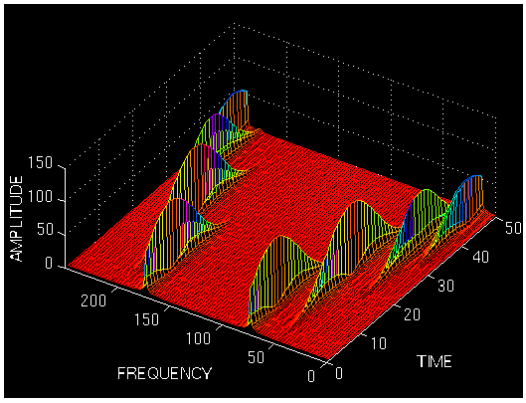


Figure 12 Result when applying Large Window

One possible solution is to use different window size for different types of signals: high-frequency signals frequently change in time, which requiring high time resolution and smaller window, whereas low-frequency signals change slowly in time, which requiring higher frequency resolution and larger window.

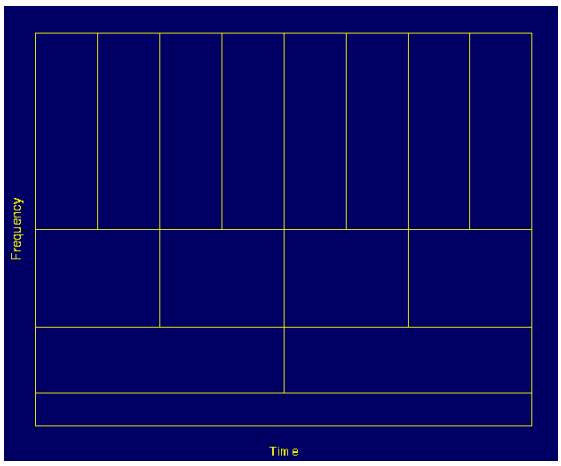


Figure 13 Dynamic Window in Wavelet Transform

To achieve this, a method called **Wavelet Transform** is introduced. The equation of wavelet is shown below:

$$f(t) = \sum_{k} \sum_{j} A a^{j} \phi(a^{j} t - kt)$$

where j is the scaling factor and k is the shifting factor. Instead of using non-fading cosine wave to decompose the signal like Fourier Transform, Wavelet Transform use wavelet which will **fade** will time and its scaled and shifted versions to do the decomposition:

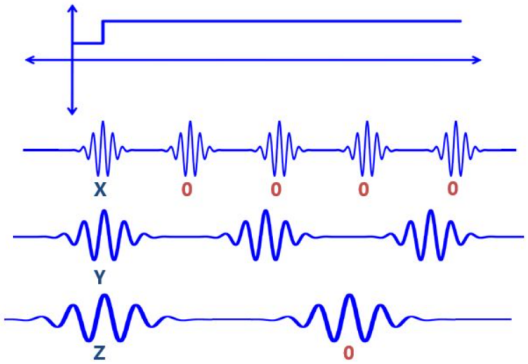


Figure 14 Example of Decomposition using Wavelet Transform

As the wavelets are finite duration, this enable them to be scaled and shifted. The extra shifting ability gives the Wavelet Transform another degree of freedom in time, compared to Fourier Transform. It can also be viewed as the combination result of window cosine bases in Fourier Transform. Therefore, we are able to locate time at the same time. Additionally, with the scaling factor, we automatically scale the window size for different frequencies. When the frequency of one base is high, the window size will decrease. Therefore, we achieve the **dynamic windowing** using Wavelet Transform

### 3.5. Smoothing

To remove the glitches, we apply filters to smooth the data. The choice of the filter is made on case-by-case basis. For the general cases, median filters are fine, and for the cases where only low frequency glitches or high glitches appear, moving maximum filter and moving minimum filter are used to achieve better results. Besides, the parameter for filter need to be tuned carefully as well to optimize the result.

### 3.6. Evaluation

We define two evaluation criteria: detection accuracy and detection error rate.

**Detection accuracy** is defined by the ratio between the number of correctly detected notes and the number of total notes. The detection accuracy can be calculated by hand as all the notes are presented in the music sheet.

$$Accuracy = \frac{n}{N}$$

where n are the correctly detected notes and N are the total notes.

**Detection error rate** is defined as the ratio between the number of correctly detected samples and the number of total samples. For detection error rate, as Garage Band is able to generate notes with exact duration, we are able to obtain a precise time-stamp standard to compared with the detected samples.

$$Error\ Rate = \frac{m}{M}$$

where m are the correctly detected samples and M are the total samples.

# 4. Results

### 4.1 Time Domain Method 1 – ACF

The detection result using ACF method is shown below:

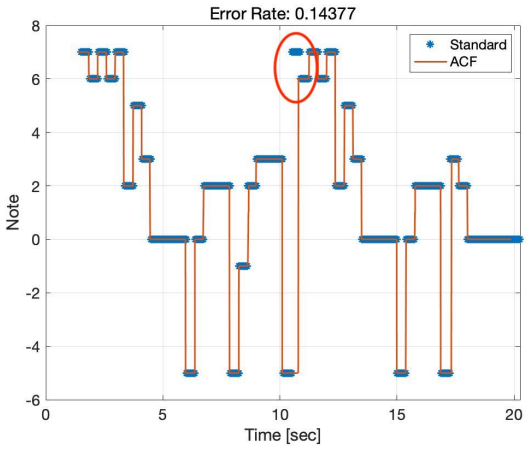


Figure 15 Detection Result for ACF

We can see from the figure that, first the silence checking is successful. The beginning silence is successfully detected. Second, the general detection accuracy is very high about 32/33, with 1 missing note. This octave jump error will be explained later in the extension section. The error rate is relatively low around 14.38%. Third, some notes may not be perfectly aligned due to the framing.

### 4.2 Time Domain Method 2 – AMDF

The detection result using AMDF method is shown below:

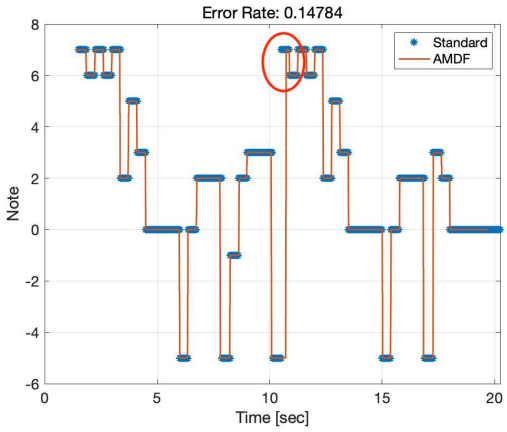


Figure 16 Detection Result for AMDF

We can see from the figure that, first the silence checking is successful. The beginning silence is successfully detected. Second, the general detection accuracy is 1 which is slight better than the ACF. This corresponds with our previous analysis. The error rate is relatively low around 14.78%. Third, some notes may not be perfectly aligned due to the framing as well.

### 4.3 Frequency Domain Method 1 – HPS

The detection result using HPS method is shown below:

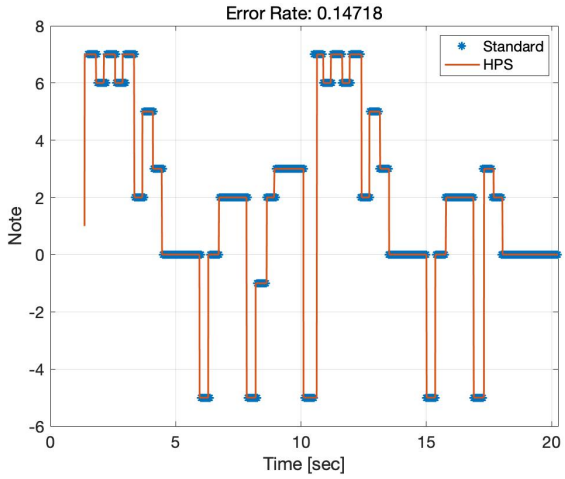


Figure 17 Detection Result for HPS

We can see from the figure that, first the silence checking is slightly unsuccessful. The beginning silence is detected except for a little drop. Second, the general detection accuracy is 1 too. The error rate is relatively low around 14.72%, slightly better than AMDF. Third, some notes may not be perfectly aligned due to the framing as well, for example in the octave jump.

### 4.4 Frequency Domain Method 2 – Cepstrum

The detection result using Capstrum method is shown below:

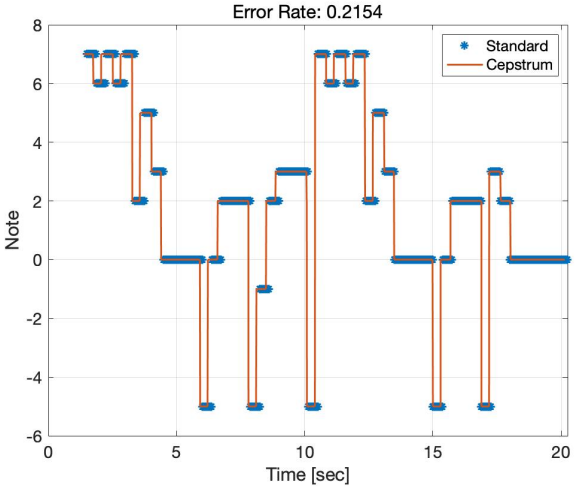


Figure 18 Detection Result for Cepstrum

We can see from the figure that, first the silence checking is successful. The beginning silence is successfully detected. Second, the general detection accuracy is 1 as well. However, the problem of octave jump is much better. The error rate is relatively low around 21.54%. Third, first few notes may not be perfectly aligned due to the framing.

### 4.5 Time-frequency Domain Method - Wavelet Transform

The detection result using Wavelet Transform is shown below:

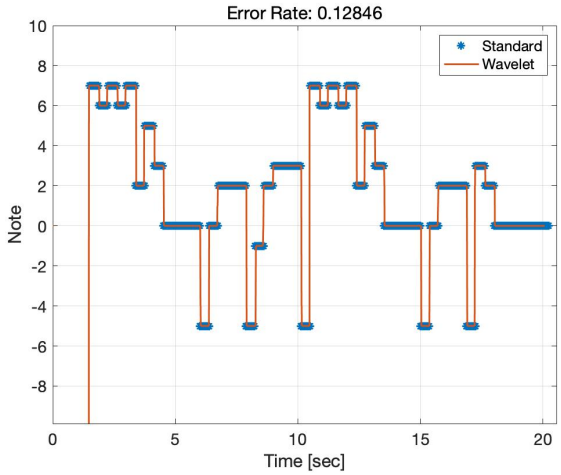



Figure 19 Detection Result for Wavelet Transform

We can see from the figure that, first the silence checking is still successful with a slight drop at the very beginning. This may be resolved into adjust the energy threshold for the silence checking. Second, the general detection accuracy is 1 as always. The error rate is the lowest around 12.85%. Third, notes are mostly aligned with the standard notes.

### 4.6 Results Comparison

The results for all 5 methods are shown below:

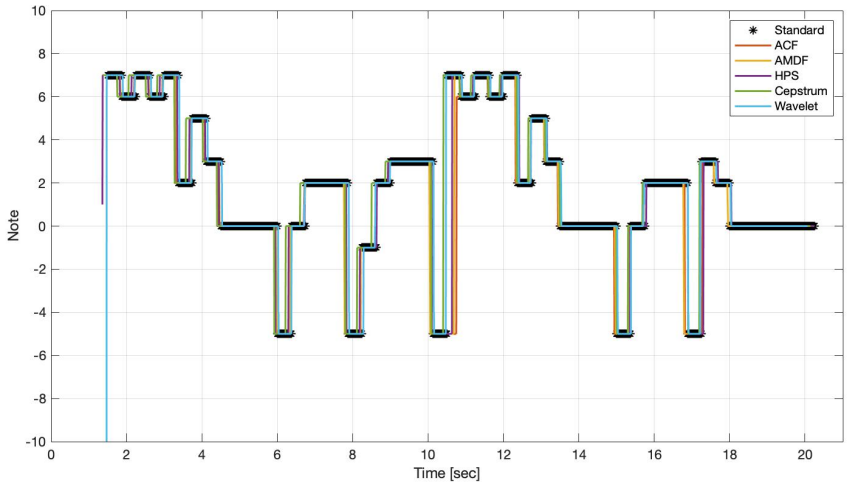



Figure 20 Results Comparison for 5 Methods

We can see from the figure that all 5 methods demonstrate very good result in terms of the detection accuracy and error rate. They primarily differ in the silence checking at the beginning and the octave jump problem in the middle. In general, ACF, AMDF, and Cepstrum have perfect silence checking, and Cepstrum and Wavelet Transform has less error in octave jump problem. In terms of these two factors, Cepstrum is the best method.

The detailed error rates of each method are shown below:

| **Method** | ACF | AMDF | HPS | Cepstrum | Wavelet Transform |
|---|---|---|---|---|---|
| **Accuracy** | 32/33 | 1 | 1 | 1 | 1 |
| **Error Rate (%)** | 14.38 | 14.78 | 14.72 | 21.54 | 12.85 |

Table 1 Accuracy and Error Rate for 5 Methods

We can from the table that all methods have perfect accuracy except for ACF. For error rate, Wavelet Transform has the lowest error rate around 12.85% and Cepstrum has the highest error rate around 21.54%. In a word, Cepstrum has the best general performance in accuracy and silence checking. However, in details, it has the largest error rate. Therefore, for different methods we have different tradeoff for silence checking, accuracy, and error rate. We may choose different methods according our specific requirement respecting to these three factors.

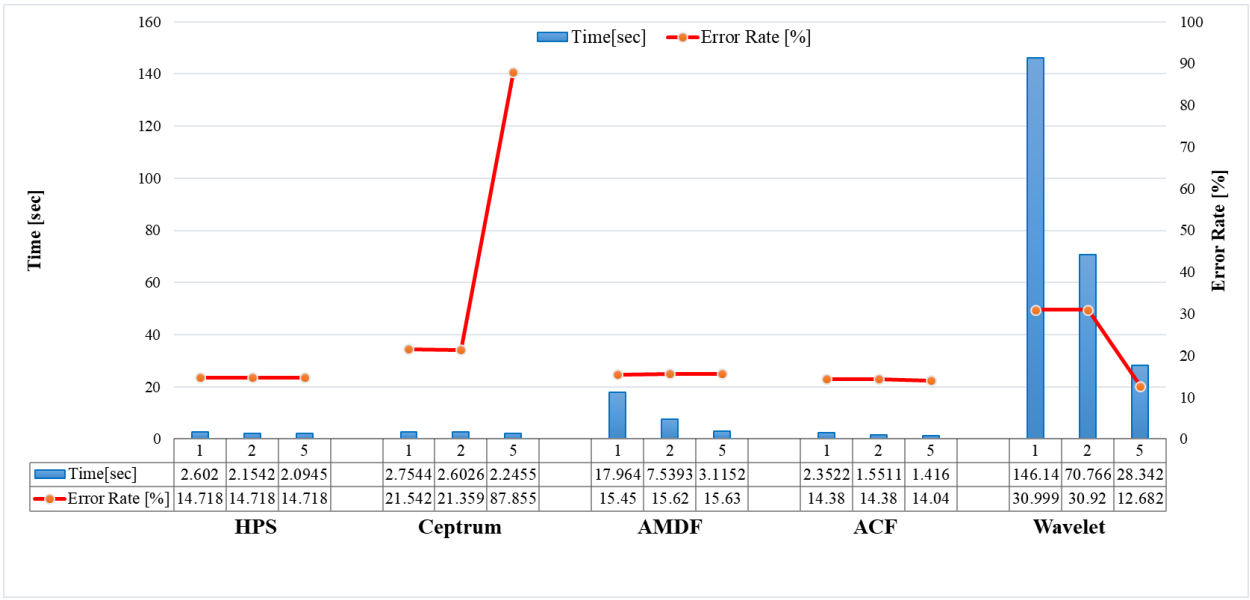
# 5. Extension

## 5.1. Down-sampling

For real application, **computation complexity** is also an important factor to consider. Shorter running time allows algorithms to perform in real time, which is definitely better. Therefore, we come up with the idea of down-sampling to reduce the computation complexity.

As we know that the frequency generated by piano are less than 4300Hz, our music would contain lots of redundant high frequency components due to the sampling frequency of 44100Hz. Therefore, we can use down-sampling to remove all the redundant information at high frequencies and shrink the number of samples to reduce computation complexity. However, the down-sampling process also take time as we need to apply anti-aliasing filter before we resample the music with a lower frequency.

The following figure shows the comparisons of time and error rate with different down sample rates for all the methods we used. The number 1, 2, 5 represent the down sample rate. (ex. 2 means down sample by factor of 2)




| | HPS | | | Ceptrum | | | AMDF | | | ACF | | | Wavelet | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 5 | 1 | 2 | 5 | 1 | 2 | 5 | 1 | 2 | 5 | 1 | 2 | 5 |
| Time[sec] | 2.602 | 2.1542 | 2.0945 | 2.7544 | 2.6026 | 2.2455 | 17.964 | 7.5393 | 3.1152 | 2.3522 | 1.5511 | 1.416 | 146.14 | 70.766 | 28.342 |
| Error Rate [%] | 14.718 | 14.718 | 14.718 | 21.542 | 21.359 | 87.855 | 15.45 | 15.62 | 15.63 | 14.38 | 14.38 | 14.04 | 30.999 | 30.92 | 12.682 |

Figure 21 Comparison of time and error rate with different down sampling rate

We can see that for all the methods, down-sampling helps to reduce the time needed for calculation, especially for methods that are time consuming like AMDF and Wavelet Transform. As for the error rate, except for Cepstrum and Wavelet Transform, the error rate almost remains the same as the we increase the down-sampling rate for other 3 methods. The error rate drops significantly as the down-sampling rate reaches 5, the reason we guess is that the down-sampling procedure helps to remove the high frequency noise and therefore achieve a better result. However, for Cepstrum whose error rate grow dramatically as the down-sampling changes from 2 to 5, the reason may be more complicated.

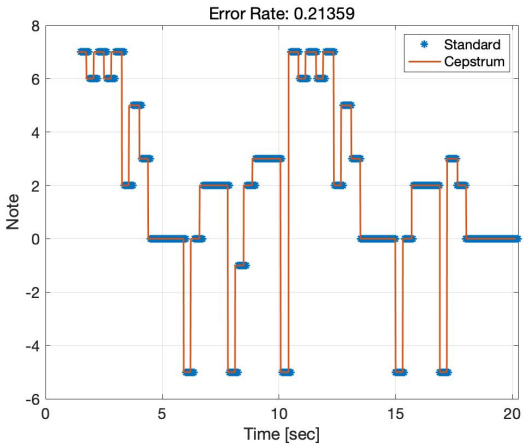



Figure 22 Cepstrum with Down-sampling Factor 2

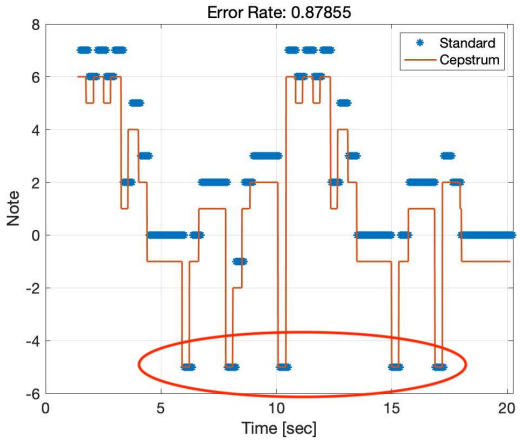



Figure 23 Cepstrum with Down-sampling Factor 2

We can see that when down-sampling rate increases, **high frequency components** are more likely to have error than **low frequency components**. This is because if we apply down sampling, then the time resolution will decrease as the sampling rate decreases, which means that the time per delay will increase. However, compared to lower frequency, higher frequency has smaller period which requires high resolution in delay (shorter time per delay) to detect correct peak. Therefore, the down-sampling will bring more peak detection error to high frequency component, and the found delay index for high frequency component is more likely to have error. Secondly, the transformation equation from index (delay) to frequency is $f = \frac{F_s}{index}$. As for high frequency component, the index will be very small, then the influence of index error will have larger impact to the final frequency calculation. With these two explanations, we are able to explain why the high frequency note are more likely to be distorted after down-sampling.

### 5.2. Octave Jump Problem

In our detection results, we found that the 18th note E5 is always missing from detection except in Cepstrum Method and Wavelet Transform. As this note is an octave higher to the previous note, we define this problem as **Octave Jump Problem**. We guess this problem is caused by the residue of previous note. As the fundamental frequency of the previous note is exactly the half of the fundamental frequency of the current note, their **overlapping harmonics** may cause interfere.

According to our guess, we proposed the **subtraction method**: subtract the residue of previous note from the current frame.

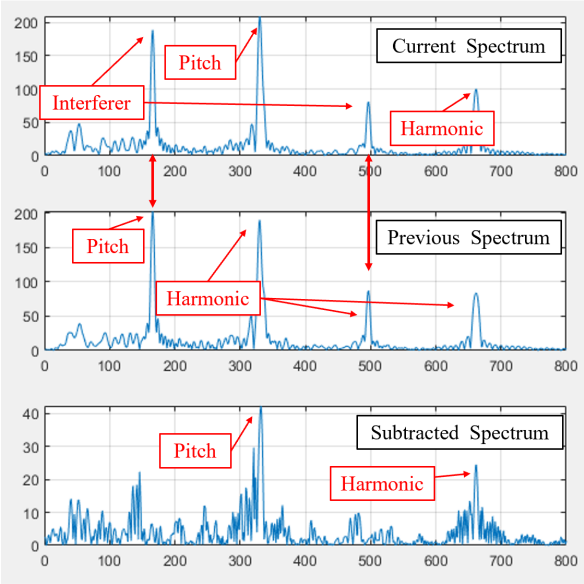



Figure 24 Process of Subtraction Method

However, this method is not applicable to time-domain methods. Therefore, we implement it in the frequency domain method HPS. The result is shown below:

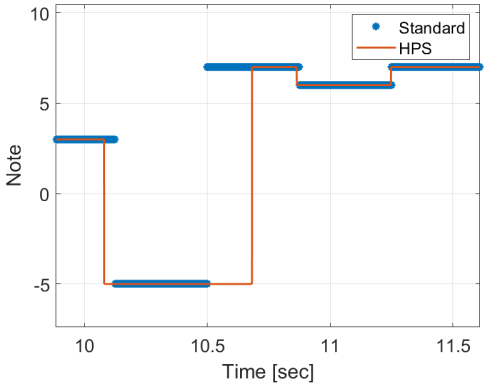



Figure 25 Result without filter and Delay

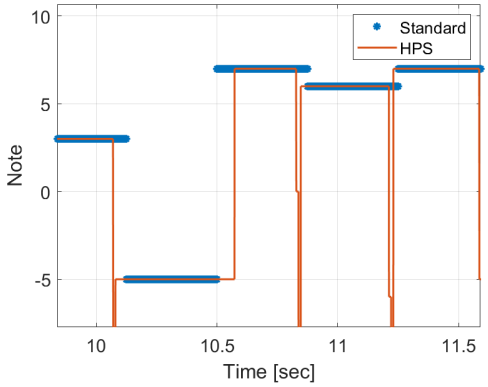



Figure 26 Result without filter but with Decay 0.86

We can see from the figures that do solve some of the octave jump error, but at the same time bring some glitches at the moment when the note changes. In experiments, we found that the number of glitches will increase as we enlarge the decay factor. However, like what we have mentioned in smoothing section, fine-tuned filters can be used to remove these glitches so the overall accuracy can be improved.

# 6. Conclusion:

In this project, we successfully developed a note detection system for monophonic music played by piano using 5 different methods in time, frequency, and time-frequency domain and evaluated their performance. Altogether these methods are presented with detection accuracy all above 32/33 and detection error rate around lower than 22%. To obtain better results, we also tried to apply down-sampling method to reduce computational complexity and subtraction method to avoid octave jump error. Besides, with further improvements such as end-point detection and polyphonic music note detection being implemented, we believe that the performance of our system can be further improved. In this project, we put knowledge we learnt in the class into practice and built a better understanding in digital signal processing.

# 7. Further improvement:

There are also some further improvements that we may continue in further research: end-point detection, weak note detection, and polyphonic music note detection.

**End-point Detection (端点检测)** is to detect the start point of each sound. The benefit of this method is that we can fix the time misalignment caused by framing. With clear separation of note, we are able to cut the extra part or interpolate the missing part to increase the detection accuracy. Another benefit is that we may cut the music into different notes interval then perform indivual detection to improve detection accuracy.

**Weak Note Detection** is the second possible improvement. In our music, we found that some notes may have smaller intensity than the other notes, which may suffer more severe interference than other notes. Our possible method is firstly applying silence checking then amplifying the weak notes by energy normalization. With energy normalization, we are able to scale the energy to a standard level and detect the weak notes.

The final improvement is **Polyphonic Music Note Detection**. It may be noticed that the music we use only has 1 or less sounding note in a certain time interval. This type of music is called Monophonic Music. In real music pieces, it is often the case that there are 2 or more sounding notes at the same time, especially for piano as it has two hands performing. This kind of music is referred as polyphonic music.

As the monophonic note detection methods we introduced above can detect notes by finding period, they can only detect one note in the time interval. The solution to deal with polyphonic music is to apply **iterative subtraction**. The detection process is very similar. We first perform silence checking to check whether there

is sounding notes. Then we perform monophonic note detection and subtract it and its harmonics from the original wave until the energy of the original wave is below a threshold, which means there is no other notes to be detected.

However, this solution also has many problems to solve. First, the amplitude relationship between pitch and harmonics of a note is different for different notes and different instruments. We need to identify the harmonics model so to subtract it from the original waves. Second, it is possible that the subtraction will do harm to the harmonics of other notes. For example, if we have two notes with frequencies being 3Hz and 4Hz, then 12Hz will be the 3rd harmonics of the first note and the 2nd harmonics of the second note. If the first note is detected and subtracted from the original wave, then harmonics of the second note will also be influenced.

# 8. Reference:


**1. https://en.wikipedia.org/wiki/Fundamental_frequency**

**2. http://mirlab.org/jang/books/audiosignalprocessing/**

**3. http://users.rowan.edu/~polikar/WTtutorial.html**

**4. Pitch and Voicing Determination of Speech with an Extension Toward Music Signals**

**5. https://zhuanlan.zhihu.com/p/44217268**

**6. https://zhuanlan.zhihu.com/p/22450818**