DSP course project: Pitch recognition and music player

Yuxi Zhang 5110309051
Ruotian Luo 5110309069
Dingjie Xu 5110309750

IEEE class, Shanghai Jiao Tong University

1 Problem Background

Many people like listening to music at the spare time. Sometimes, they want to know the score or the pitches of the music, but it is difficult to figure out the pitches for an amateur. On the other hand, some music fans want to play the music on the computer with different timbres. Based on this situation, our group has designed a compound music software to realize these ideas.

2 Project Task

1. Recognize and show the pitches in a piece of music synchronously;
2. Offer users a music player: a note player with virtual piano keyboard to play single note, and a score player to play a piece of score written by users. Users can choose to play the music with either piano timbre or violin timbre.

3 Pitch Recognition

3.1 Shot-Time Fourier Transform

We want to deal with the music signal, so we need to analyze it in the frequency domain. Since the signal is continuous and lasts long time, we decide to use Short Time Fourier Transform to handle it slot by slot.

Short-time Frequency Transform (STFT) is a transformation which is used to determine the frequency and phase information of continuous signal. We first choose a window function w(t), which is stable in a short time slot, to get the power spectrum of the signal. In this case, x(t)w(t) should also be stable in the time slot. Equation 1 shows the mathematical form of Short-Time Fourier Transform.

$$x(\tau, \omega) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-j\omega t}dt \tag{1}$$

After taking STFT, we can obtain the frequency spectrum in each time window.

3.2 Harmonic Product Spectrum

In music, each pitch has a base frequency and a series of harmonic frequency, which is called overtone. Different pitches have different overtone ratio, and the base frequency is not necessarily of the largest amplitude, which brings us difficulties to recognize the base frequency of the pitch in the frequency spectrum.

A simple idea is to regard the first peak C the peak with lowest frequency C as the base frequency. This method works well if the music is single tune and with little noise in the frequency spectrum. However, when it comes to compound tune, this method cannot recognize all the notes. In another case, if the rhythm is very fast, leads to some noise in the spectrum, it is hard to find the first peak, the real peak of the pitch rather than the noise.

David has summarized several pitch recognition methods in his book *Pitch Extraction and Fundamental Frequency: History and Current Techniques*[1]. After trying different pitch recognition methods, we adopt the Harmonic Product Spectrum algorithm, which is presented by Patricio[2].

The HPS algorithm involves two steps: down-sampling and multiplication. To down-sample, we compressed the spectrum twice in each window by resampling: the first time, we compress the original spectrum by two and the second time, by three. Once this is completed, we multiply the three spectra together and find the frequency that corresponds to the peak (maximum value). This particular frequency represents the fundamental frequency of that particular window. Figure 1 shows the float chart of this algorithm.

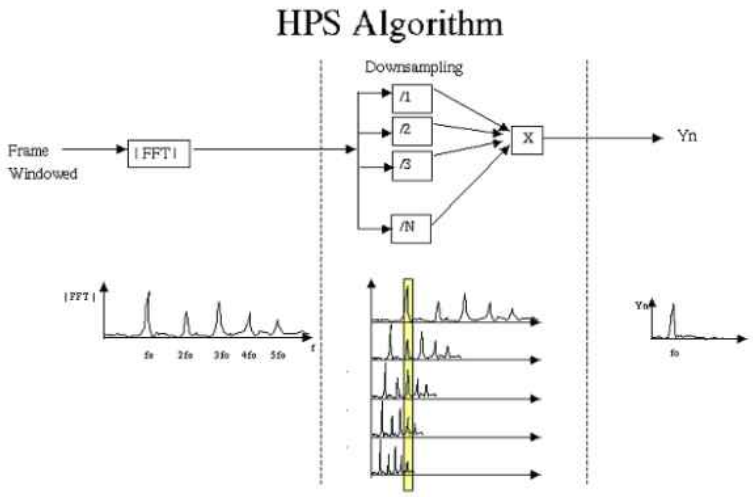

Fig. 1. First, the windowed frame is taken into the frequency domain and the magnitude of the spectrum is calculated (left). Next, the spectrum is downsampled to create more compressed versions of itself (center). Notice how the higher harmonics of the fundamental frequency align with each other in the downsampled spectra. Last, a multiplication of these spectra is performed and the maximum is found (right). This corresponds to the fundamental frequency.

After this algorithm, we just need to take a peak detection algorithm to extract the base frequency. In addition, with Equation 2, we can transfer the frequency value to the pitch value.

$$P = 69 + 12\log_2(\frac{f}{440}) \tag{2}$$

Here pitch value is MIDI standard.

3.3 High Frequency Amplification

In HPS algorithm, there is a defect. The high frequency components of a pitch are often very small. If we directly multiply them together, the base frequency will be eliminated due to the small amplitude in high harmonic components. Therefore, we present a *High Frequency Amplification* (HFA) algorithm to overcome this problem.

In HFA algorithm, we do a preprocessing on the frequency spectrum before HPS algorithm. Our purpose is to amplify the high frequency part, therefore we take the amplitude times the frequency value as Equation 3 shows.

$$X(f) = X(f) * f \tag{3}$$

After this HFA preprocessing, the HPS algorithm can perform better.

3.4 Implementation

In implementation, we take hanning window with length 0.1s. In other words, we recognize the pitches every 0.1s. We take the famous violin music *butterfly* as the test sample.

In STFT part, we can use the Matlab function *spectrogram* to get the STFT of the signal as following.

```
[S F T P] = spectrogram(x, window, Noverlap, nfft, fs, 'yaxis');
```

Here x is the music signal and window is the window function that we use to sample the signal. Here we use the default window Hamming window. And Noverlap is the overlapped sample in each time slot. The default value is 50%, which means there will be 50% overlap in each sample. We set it to be 0 and in this case, there will be no overlap in each sample. Nfft is the length of FFT transformation. To make the result more precise, we set the value nfft to be 8192 here. Ft is the sample rate.

The result S is the STFT of x. F is the frequency vector that we use in the input. T is the time point that we take sample and P is the power spectral density.

In HPS algorithm, we take down-sampling process 4 times and figure out the product. This algorithm have a good perform when the music is not very complex. Figure 2 shows the process in a certain time window.

From the figure, we can see that HPS algorithm and HFA algorithm have a good performance on the real music.

4 Music player

Another task of our project is using limited resource to generate the pitches as many as possible. Here we separate the task into two parts: single-tone generation and score generation.

4.1 Single-tone Generation

Objectiveness The objectiveness of single-tone generation is to using the resources we've gotten to generate the tones of different duration, pitches and timbres.

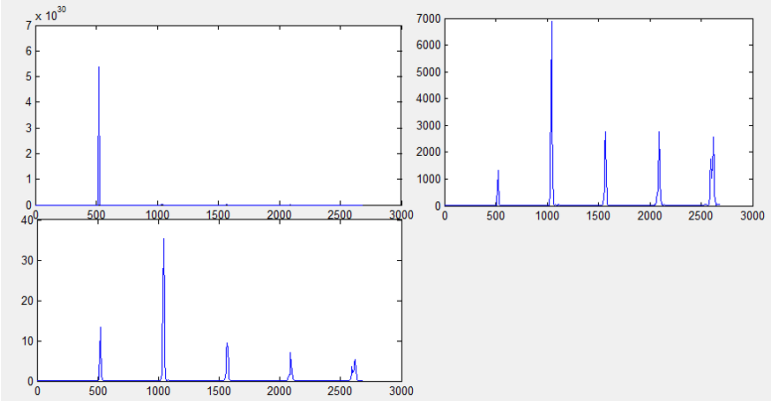

Fig. 2. HPS algorithm results: The left-top figure is the final result. The left-bottom figure is the original spectrum. The right-top figure is the spectrum after HFA algorithm

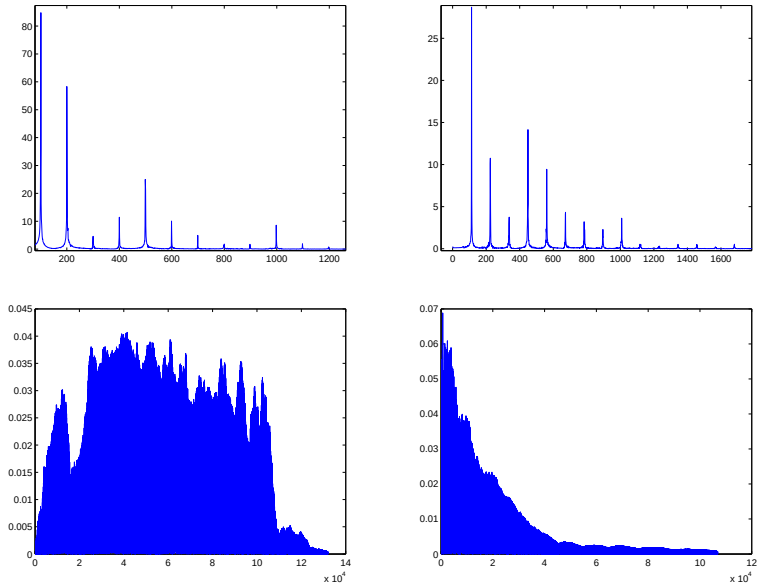

Fig. 3. Frequency domain of same timbre and the time domain of different timbres. (The up two are the frequency distributions of violin in C4 and D4, the down two are the signals of violin and piano in C4.)

Idea By evaluating the frequency domain and time domain of tones of different musical instruments, we found that for the same musical instrument, the frequency distribution of base frequency and harmonic frequency is different of different pitch. Besides, the timbre of a certain pitch is not only decided by its frequency distribution, but also the envelop of the signal. So, aiming to get the tones real enough, we first manually record the pure tones of piano and violin of a certain duration from C4 to B4. And using the raw material, we hope to get tones of different duration and other pitch.

Change the pitch Actually this part is very easy, we only need to interpolate the signal of other frequency and get a good pitch.

Change duration As discussed before, we know the envelop of signals are important, so we need to check the envelop of piano and violin.
The piano envelop is very simple: it attenuate regularly. So to get piano tone of different duration, we only need to cut off or pad zero in time domain.
The envelop of violin is quite different, it has three states: begin, middle and end. All the change on duration can only be done on the middle part. So given the duration we want, we duplicate or cut the middle part to the length we want.

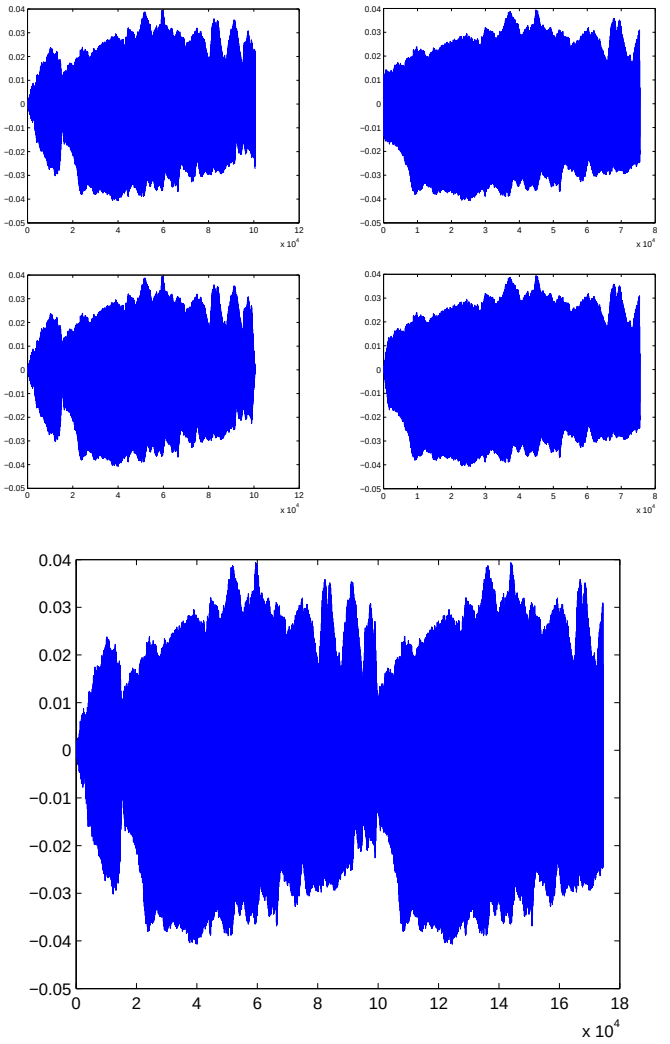

Fig. 4. The link of two signals. (The first row is the given two signals. The second row is the processed two signals. The third row is the result.)

Kick out the burst However when we are doing cut or duplication naively, we can hear a harsh burst in the connection point. At first, we want to eliminate the burst by using a low-pass filter. However after using the low-pass filter, the timbre is not the same. Finally, also from the envelop intuition before, we process the signal in time domain. When we are linking two signals together, we let the end of the first part attenuate to zero gradually, and let the beginning of the second part arise from zero, and then let the tail of the first signal and the head of the second signal add overlappingly.
Similarly, in the end of a signal is not smoothingly attenuate, there will also be a burst. So, in this scenario, we simply let the tail attenuate.

4.2 Score generation

After getting the result of the former part, it's actually easy to realize the score generation. Given the duration and pitch of each tones, and using the "connect" function, we only link the tones together, and thus getting a track. If we have another track, we just add the two tracks point-to-point.

5 User Interface

According to our project purpose, we design the User interface as Figure 5.

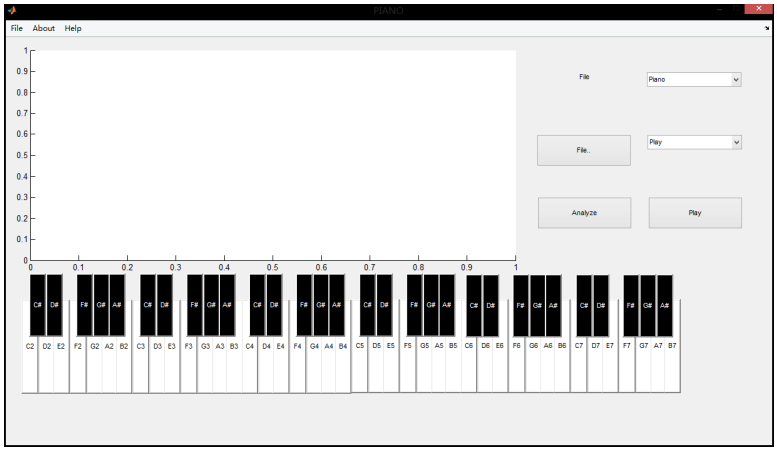

Fig. 5. User Interface

The main part of the user interface is a diagram, which is used to show the frequency spectrum. We can see in each time slot, which frequency has the highest amplitude and based on our observation, we can get the base note and then high light it on the board. To load the file, we design one button file. When we click the file button, we can choose a file to upload. Here we use the function *uigetfile()* to get the name of the file we choose. The function is shown below:

```
[filename, pathname] = uigetfile ('*.wav','Pick an wav-file');
```

The default file type is *.wav and once we choose a file, we can get the filename and the path of the file. We just show the path in a static text box. Since we want to analyze music signal, if we try to load one file which is not wav type, it will show Error box.

Click the analyze button, we can deal with the signal. Use openfile() function to get the music file and then use wavread() to get the original signal.

To avoid confusion, here we also create two drop-box to choose play music or analyze signal. If we choose play music, we can click the keyboard to call related function to play the key. And we can also choose violin mode or piano mode. But if we choose analyze mode, then the keyboard will not respond even when we click the key.

Here we also give an example. You can click the button Play and you will hear the music canon which is composed by us by combining the waveform of related keys.

Whats more, you can see related information if you click Help or About.

6 Future Work

There are mainly two aspects we can do in the future:

1. The HPS algorithm sometimes do not perform well when the music is very complex and with two or more pitches at the same times, especially when the frequency of one pitch is just the times of the frequency of another pitches. In this situation, this algorithm can hardly distinguish these two pitches. We plan to model on the ratio of the harmonic components and build a generative model to solve this problem.
2. Given a piece of music, we want to transfer its playing instrument. For example, given a piano version *Cannon*, we want to transfer it to a violin version. This part is not very difficult if the pitch recognition part is perfect. We just need to transfer the recognized pitches into a piece of score, and then generate a new piece of music with another timber according to the score. The second step is already done in our current work.

7 Contribution

Name	Responsible part	Contribution
Yuxi Zhang	Pitch Recognition	33.3%
Ruotian Luo	Music Player	33.3%
Dingjie Xu	Project Design and UI implementation	33.3%

References

1. Gerhard, D.: Pitch extraction and fundamental frequency: History and current techniques. Regina: Department of Computer Science, University of Regina (2003)
2. De La Cuadra, P., Master, A., Sapp, C.: Efficient pitch detection techniques for interactive music. In: Proceedings of the 2001 International Computer Music Conference. (2001) 403–406