Music Analysis and Synthesis
Assignment 2: Fundamental

Shih-Ho Wang

March 8, 2019

1 Introduction

The target is to find the accurate frequency and the inharmonicity of every note in a sound track. The onset, duration, and velocity of notes are given.
According to the data already given, It is possible to accurately aim the target note and do the sound analysis.
First, find the exact duration of every note in the audio file from the exact start time and end time. Second, do the FFT of each duration. Since we have a rough note reference, we can use the note reference to search the nearby region to see in which frequency bin does it contain the relatively large energy, and it might be the accurate frequency of this note. After scanning all FFT in every onset in the audio track, we can count the means and standard deviations of each note.

2 Implementation

2.1 Calculate down sample ratio

To find the accurate fundamental frequencies, the low frequency side is more important than the high frequency side. In a normal audio recording, the most popular sample rate is 44100 Hz, which means it can detect up to 22050 frequencies in the audio file.However, it is too much waste of computation since the highest fundamental frequency is still much less than 22050.
Since the largest note of the reference audio file, the Bach's performance, is 81, I choose 88 to gain more margin. In normal tune, $A4 = 440Hz$, the corresponding frequency of note 88 is 1319Hz, which means the minimum required sample rate is $1319 * 2 = 2638Hz$. That means, we can change the sample rate to 2638 Hz, and let the frequency resolution be focused on the low frequency part. In this situation, the downsample ratio can be changed to $floor(44100/2638) = 16$.
Note, in the MATLAB code, it would automatically generate the appropriate down sample ratio according to the sample rate.

2.2 Calculate the number of samples for FFT

The number of samples for FFT is directly related to the resolution of frequency bins. Say if we want the resolution be 0.5 Hz, we should choose the number of samples for fft, based on the new sample rate 2638 Hz, is $frac26380.5 = 5276$. To choose a number powered of 2, it would be 8192. To conclude, No matter how long is the duration, the sample will always be processed a 8192 point FFT. If the number of sample is not enough, it would be zeropadded, to get high resolution frequency bins.

2.3 Finding frequenies based on normal tune: $A4 = 440Hz$

Finding frequenies based on normal tune: $A4 = 440Hz$ is the first approach, because, before analyze the audio file, we have no information about what the actual tuning is, which means we would update it later.

First, give a larger scanning zone, like 10%, and find the largest energy tone from FFT. The corresponding frequency is regarded as the actual frequency of the note. Admittedly, it is not a safe guess since 10% variation might include other frequencies from other notes. However, it is just the first approach.

2.4 Learning and updating real A4 frequency

Once the proposed frequency has been calculated, we can continuously calculate its error between the proposed and the theoretical frequency based on $A4 = 440Hz$. Then, we can update A4 by the error percentage, applying a learning rate to decrease the risk of instability. In this work, the learning rate is set to be 0.2, which seems good enough.
After updating A4, we are more and more confident in guessing the accurate freqeuncies. However, the scanning zone still has to be at least a small value, which is not 0, and it can scan the smaller range to find the accurate frequency. The desired minimum scan region is smaller than half of the difference between 2 semitone, which is about 2.8%. But if we are confident enough, we can choose a smaller value. In this work, we set 2% as the minimum scan region percentage. As described above, the update function and the learning curve would be:

Listing 2.4.1: Update function

```
% ------ Update A4Freq & scanPercent in each iteration ------
% Calculate difference of Freq to update actual A4Freq
errorPercent = (proposedFreq - refFreq) / proposedFreq;
% update A4Freq, update rate can also be tuned by learningRate
A4Freq = A4Freq * ( 1 + errorPercent * learningRate);
% update scanPercent by the error of this note,
% but remain 0.02 if the error is very small
% 0.02 is good enough because Note difference is still 0.0561
scanPercent = max(abs(errorPercent), 0.02);
```

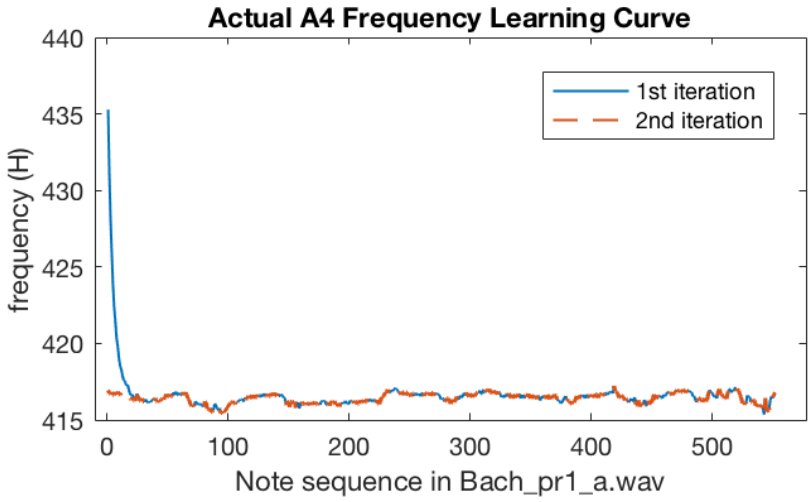

Figure 2.4.1: Actual A4 Frequency Learning Curve

Obviously, in first iteration, the learning curve is settled in 100 samples. In order to get the accurate result, we would do the iteration 2 times, and only take the second iteration data into analysis.

3 Conclusion

3.1 Result

	Bach_pr1_a.wav		Bach_pr1_b.wav		
Note	Frequency (Hz)	Variation	Frequency (Hz)	Variation	Samples
36	62.24	0.00	62.24	0.00	5
37	0.00	0.00	0.00	0.00	0
38	0.00	0.00	0.00	0.00	0
39	0.00	0.00	0.00	0.00	0
40	0.00	0.00	0.00	0.00	0
41	82.77	0.00	83.10	0.00	2
42	87.48	0.00	87.14	0.00	2
43	92.86	0.00	92.86	0.00	18
44	98.25	0.00	99.59	0.00	2
45	0.00	0.00	0.00	0.00	0
46	0.00	0.00	0.00	0.00	0
47	116.75	0.00	116.41	0.00	1
48	124.15	0.00	124.32	0.18	10
49	0.00	0.00	0.00	0.00	0
50	138.96	0.00	138.96	0.00	16
51	147.03	0.00	148.71	0.00	2
52	155.78	0.00	155.44	0.00	10
53	165.41	0.31	166.21	0.36	18
54	0.00	0.00	0.00	0.00	0
55	185.52	0.28	185.70	0.47	48
56	196.15	0.00	198.85	0.00	2
57	207.94	0.06	207.91	0.12	30
58	220.72	0.00	222.36	0.11	10
59	233.16	0.00	232.32	0.17	28
60	248.31	0.07	248.64	0.00	69
61	0.00	0.00	0.00	0.00	0
62	277.72	0.17	277.88	0.10	34
63	294.06	0.00	297.43	0.00	4
64	311.66	0.58	310.85	0.57	41
65	330.52	0.75	332.04	0.91	36
66	348.40	0.23	347.25	0.10	12
67	371.10	0.25	371.44	0.20	43
68	0.00	0.00	0.00	0.00	0
69	415.54	0.08	415.52	0.00	16
70	0.00	0.00	0.00	0.00	0
71	466.13	0.34	464.20	0.17	12
72	496.31	0.24	496.88	0.24	25
73	522.85	0.00	531.60	0.00	4
74	555.53	0.33	555.66	0.30	23
75	0.00	0.00	0.00	0.00	0
76	623.31	0.17	621.27	0.18	12
77	661.85	0.11	665.10	0.15	9
78	0.00	0.00	0.00	0.00	0
79	742.47	0.17	742.90	0.39	4
80	0.00	0.00	0.00	0.00	0
81	831.05	1.10	831.13	0.97	4

Figure 3.1.1: Result Table

As the result table above, the variation are all very small, indicated that the mechanism works.

3.2 Matlab Codes

There is a test bench called ass2_testbench.m, which will run the function fundamentals as required.

In the function fundamental.m, there is a note summary which will be automatically produced after calculating throw the audio file. However, the required format doesn't allow to output it. If anyone is curious about the summary, they can set a breakpoint at the end and inside the function, so they can saw a noteSummary just almost like Fig- 3.1.1