Investigation of Pitch Detection Characteristics from Different Audio Context

Final Project Report

Mingfeng Zhang and Shumin Xu

Abstract:

In this project we investigate the possible factors that affect the accuracy of musical pitch detection algorithms. In particular, three factors investigated include: (1) the different timbre of the same note, especially music note of different complexity; (2) overlapping frequency domain contamination, such as incomplete polyphonic separation; and; (3) concurring noise. We implemented a MATLAB toolbox for synthesize tones and use the synthesized tones and real music tones to test the pitch detection performance. Then we implemented a test environment for pitch detection functionalities. We then evaluate the pitch detection accuracy at simulated test scenarios of each factor. This study is useful for designing robust musical information processing tools for practical application scenarios.

1 Introduction

The pitch detection accuracy of current music analysis algorithms tends to degrade with imperfect audio source materials. For example, pitch detection algorithms tend to make mistakes when strong noise present. We want to investigate the extend that them interference/imperfection will degrade performance.

For practical applications such imperfect recording scenarios are prevalent. For example, if we want to design a search engineer that find similar songs from a recording the user “snapped” at a supermarket. We usually have strong ambient noise. Other factors will also affect the pitch detection accuracy. In this work we will analyze the effect of music note complexity, the interference from concurring note, and the effect of noise.

2 Test factor I: Complexity of Music Notes.

We programmed a audio synthesis algorithm that allows us to generate an audio file that simulate the music note. Because we synthesized these test tones, so we know the exact parameters of the test tones and can compare these parameters with detected pitch and detection performance. Synthesized tones also help us to form a well-controlled experiment environment.

Here we synthezied a tone of 440 Hz. We then add in some amplitude modulation and frequency modulation to each sonic partials. We synthesized 6 sonic partials. The program is at PJ_ClASS_TESTTONE_E02_Experiment_1.m, PJ_ClASS_TESTTONE_E02_Experiment_2.m, PJ_ClASS_TESTTONE_E02_Experiment_3.m.

The spectrogram of our first synthesized tone is plotted in Figure 1 and Figure 2. The MATLAB program is in PJ_ClASS_TESTTONE_E02_Experiment_1.m. Here the f0 is 440Hz. We use the pitch detection functionalities in MIR toolbox [1] to detect the audio pitch. The MIR pitch detection functionalities in embedded as:

```
audioobj1 = miraudio(E_TR1_TRANS, 44100);
[p ac] = mirpitch(audioobj1,'Mono');
notepitch = mirgetdata(p);
```

The first line package E_TR1_TRANS as an MIR toolbox formatted object. Then we run MIR pitch detection algorithms and get the pitch. More code is provided in the analysis toolbox.

Then tone is very simple so the detection result is good. Here MATLAB MIR toolbox gives the result of 440.52 Hz.

In Figure 3 aqnd Figure 4 (code at PJ_ClASS_TESTTONE_E02_Experiment_1.m), we use another synthesized tone at 440 Hz. We add in some amplitude modulation and frequency modulation to each sonic partials to add in the complexities. Here AM index = 0.5, maximum frequency deviation = 10 Hz at f1. The detection result is 443.33Hz. We can see that a higher complexity lead to some more error.In Figure 5,6 we can see a peak at 440Hz at fft plot. Here it seems human can do a better pitch detection than algorithm with the help of simplr signal analysis tools.

In Figure 7 (code at PJ_ClASS_TESTTONE_E02_Experiment_1.m), we synthesized another more complex music note.Here we change the maximum frequency deviation to 40 Hz at f1. From Figure 7 the note is more complex. The detction result is 449.77Hz. The reading has more errors because the note is more complex.

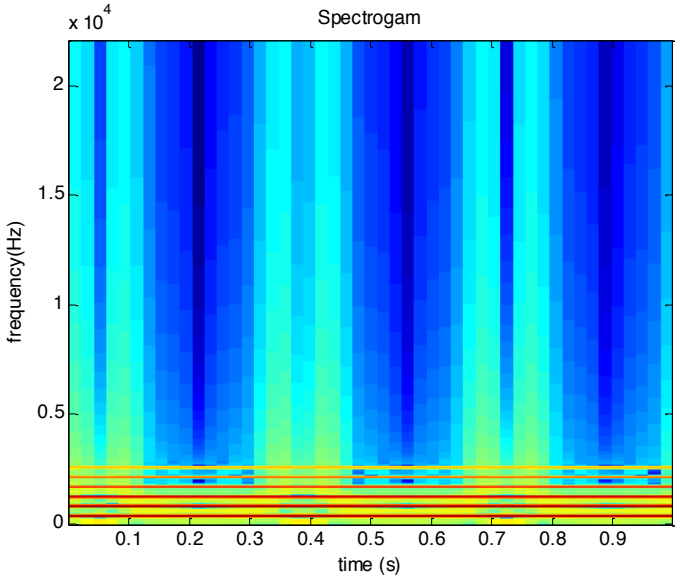

Figure 1. Spectrogram of Syntehsized Tones A

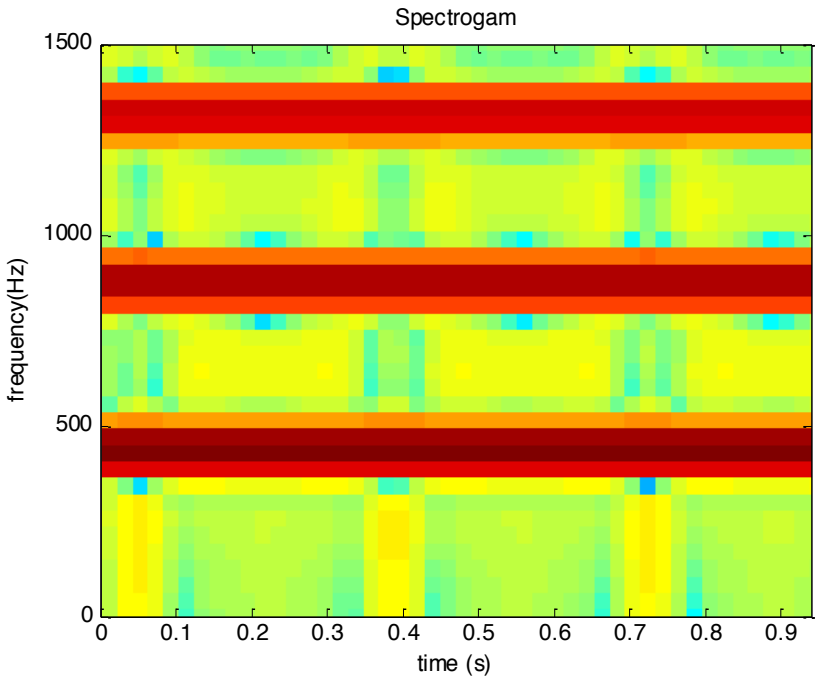

Figure 2. Spectrogram of Syntehsized Tones A, zoom in, we can see the first sonic partial is near 440 Hz.

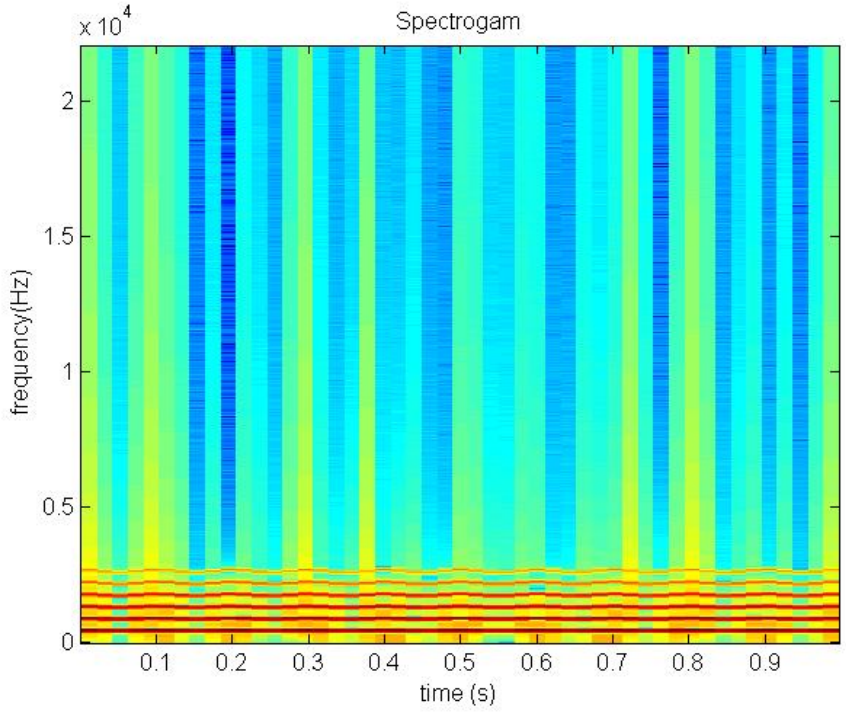

Figure 3. Spectrogram of Syntehsized Tones B. We can see that the tone is more complex. Here we see the zigzags as the resulk of frequency modulation.

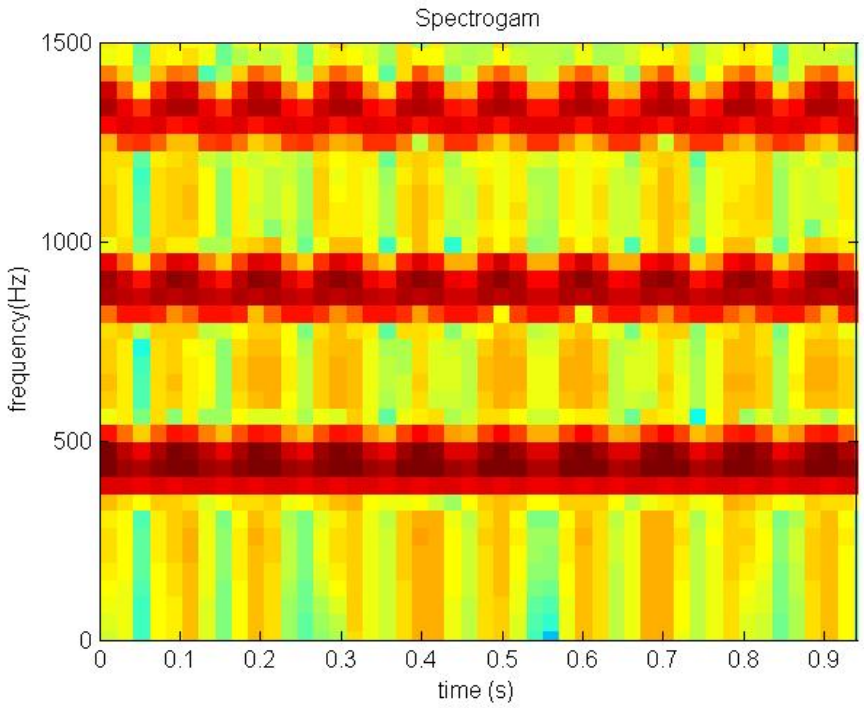

Figure 4. Spectrogram of Syntehsized Tones B, zoom in, we can see the first sonic partial is near 440 Hz. We can see that the tone is more complex

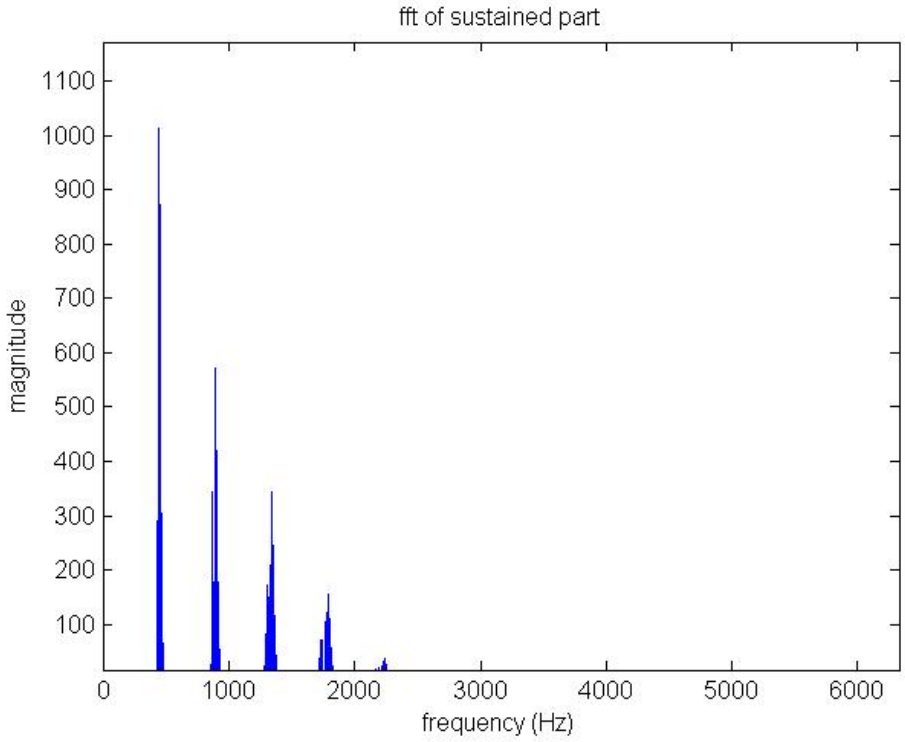

Figure 5: FFT of test tones at 440 Hz, with 6 harminic partials.

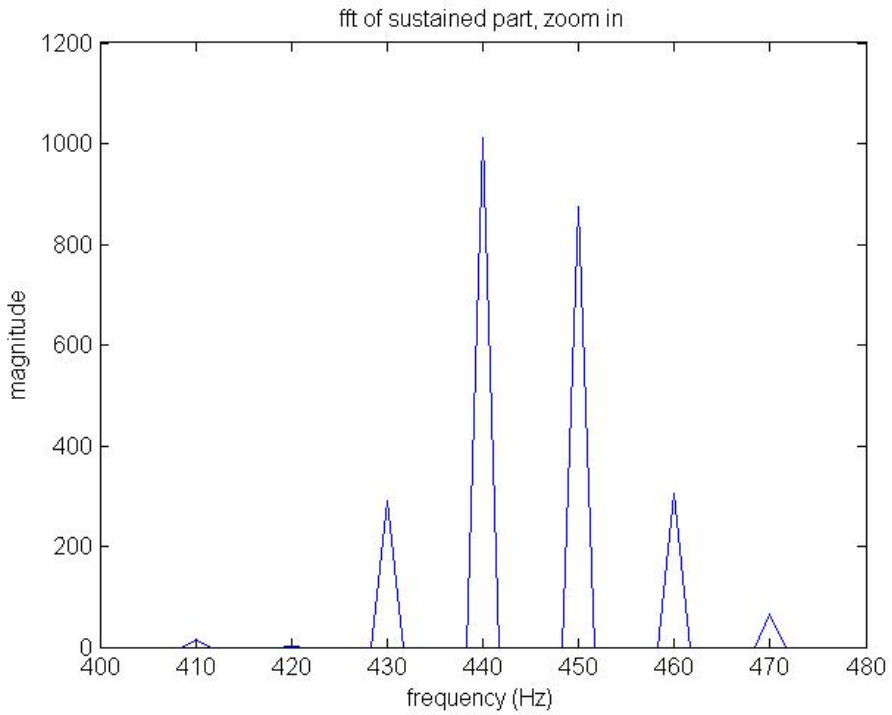

Figure 6. FFT of test tone near 440Hz. The other smaller peaks is generated from frequency modulation.

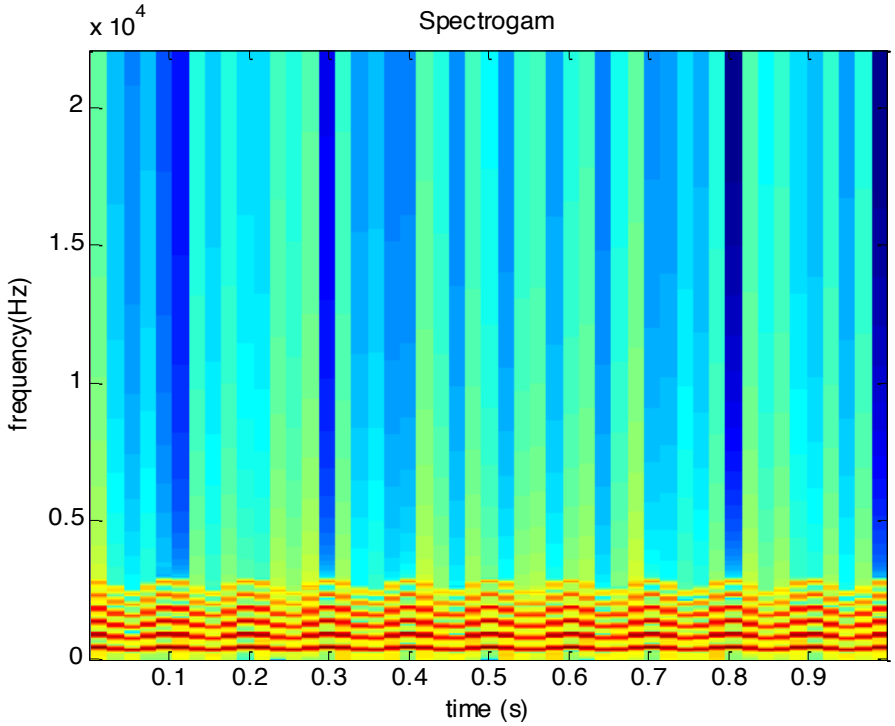

Figure 7. Spectrogram of Syntehsized Tones B. We can see that the tone is more complex. Here we see the zigzags as the resulk of frequency modulation.

3 Test factor 2: Interference from other Music Notes.

The test environment is implemented in PJ_CLASS_PITCHER_E03_Experiment_1.m, PJ_CLASS_PITCHER_E03_Experiment_2.m in the project folder.

3.1 Tests on Real Music Tones

Here we run an audio file "vibrate1_example.wav" through this test environment and shows the detection process. This audio file is a real music tone. This is an oboe music note. The MIR toolbox provided the reading of 596.75 Hz.

We first plot the audio spectrogram in Figure 8. Then we zoom in at Figure 9. From this Figure we see the pitch detection is good. More accurate frequency can be read from Figure 10 and Figure 11. The MATLAB program we implemented also runa peak picking at Figure 11 and the reading is 595.60 Hz (see the variable " `peak_freq`").

Then we run the same program on other audio file (see our toolbox), and we get similar result. So without interference, MIR toolbox perform satisfatorily.

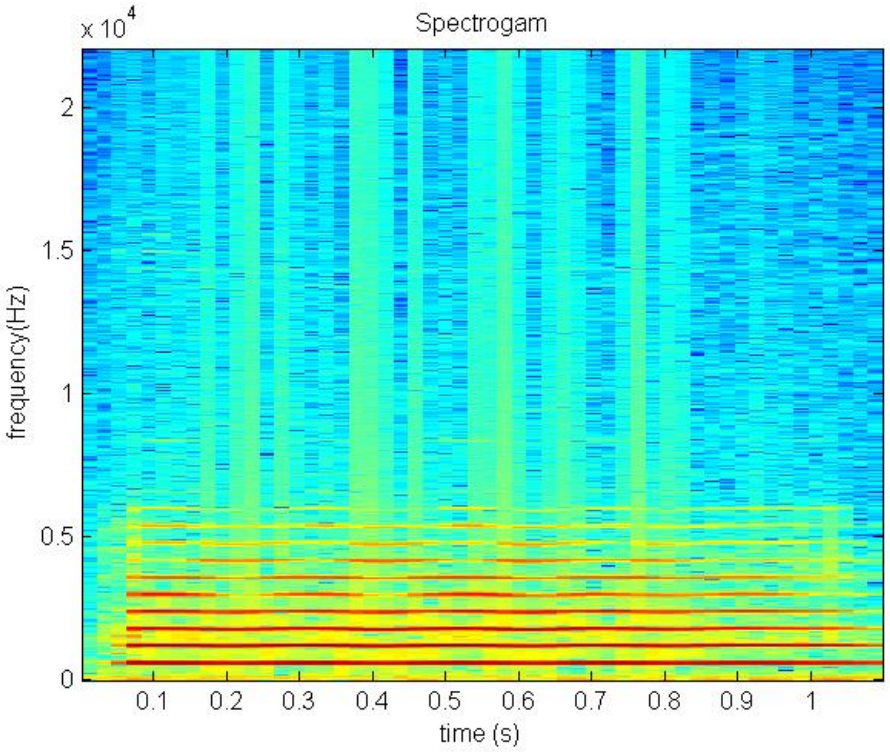

Figure 8: Spectrogram of oboe note

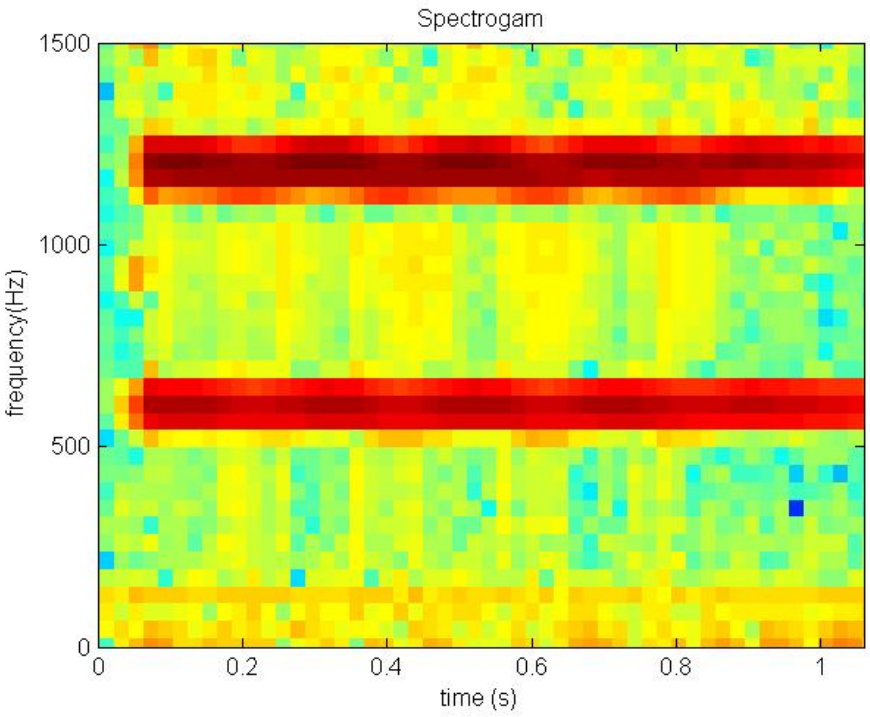

Figure 9: Spectrogram zoom in, we can see that the pitch frequency is around 596Hz.

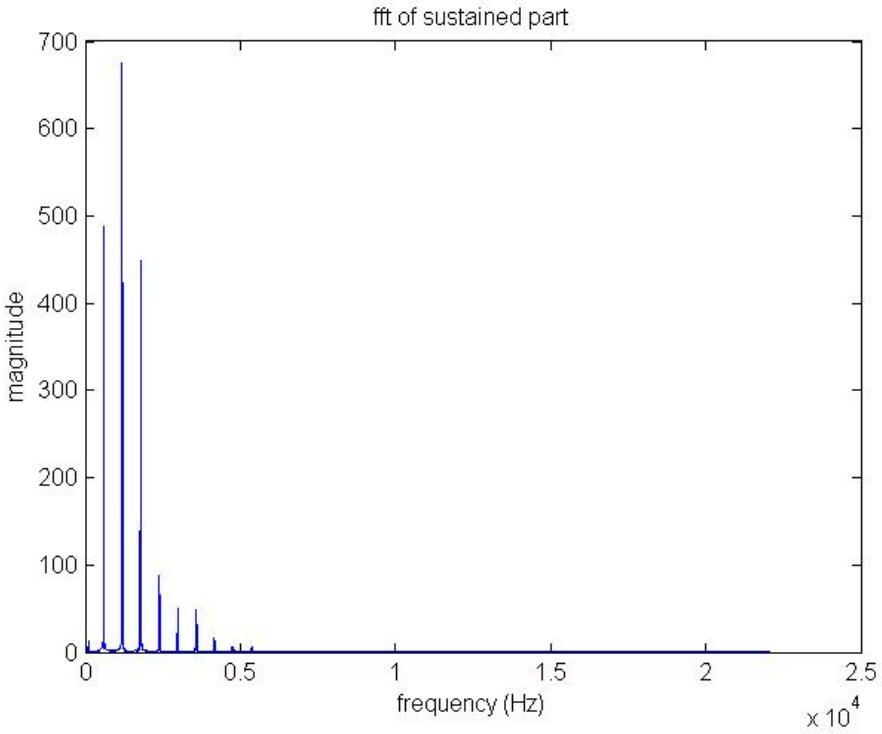

Figure 10: FFT of the sudtained part of this music note

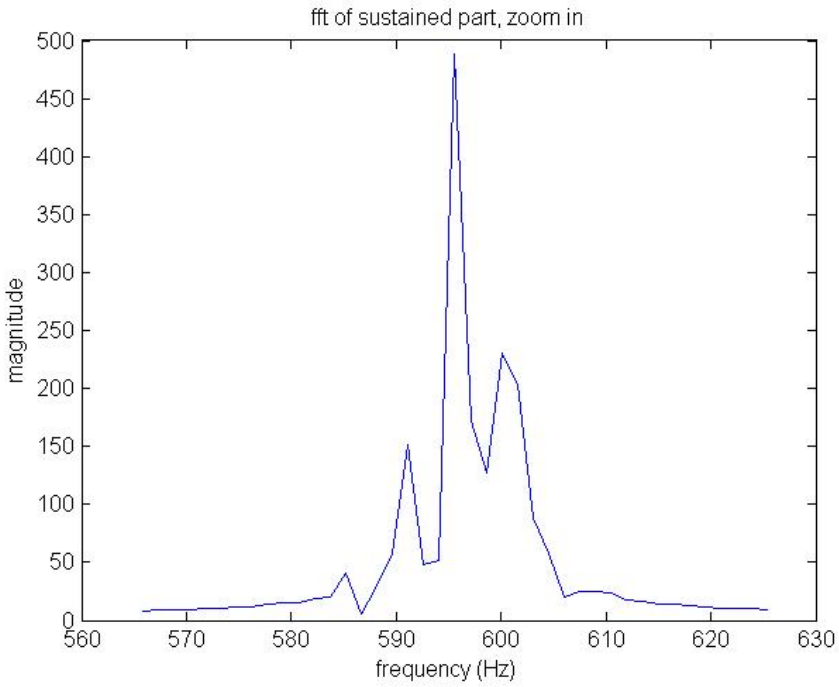

Figure 11. Detail of FFT near the pitch frequency

3.2 Residuals of Concurring Notes

The residuals from other music instrument can be added to test the performance of pitch detection algorithm. This experiment setting is to simulate two audio processing tasks.

- transciption from incomplete audio source separation
- audio recording from imcompleted isoloated emsemble. Usually we use aocsutcial walls to separate instruments during an multitrack emsemble recording, but the separation can be incomplete.

We test the pitch detection algorithm in MIR toolbox and see if it is interference robust. The setting is the same with Sec. 2.1. We use a framework to add in another note. Because this note is interference note, the level is set much lower. Here the SNR signal noise ratio is 3.5 dB. This note is a violin note at fundamantal fruency of 1041Hz. We combined these two notes and run detection again. Here the detection algorithm makes a mistake. It should be the same with the first note at around 597Hz (see the spectrogram in Figure 12) because it is the dominate note. But here it made a mistake.

From the fft of combined note in figure 13 we say a dominate fundamental frequency at 597Hz, it seems that we can detect it nicely. but this allgorithm cannot. Sometimes human is better than computer.If we zoom in the fft of combined note, we see another significant peak at 1040 hz, the

MIR detection result seems to be near the interference note The main audio frequency is 596Hz. Here the MIR pitch detection algorithm made a mistake and give a reading of 1034 Hz. Around the frequency of the concurring residual note.

We can adjust the SNR and see the detction results. The result is plotted in Figure 14, 15. We can observe a jump at 5db, where the MIR detection result jump to another note. The source code for this batch analysis is provided at PJ_CLASS_PITCHER_E03_Experiment_2_morepoints.m.

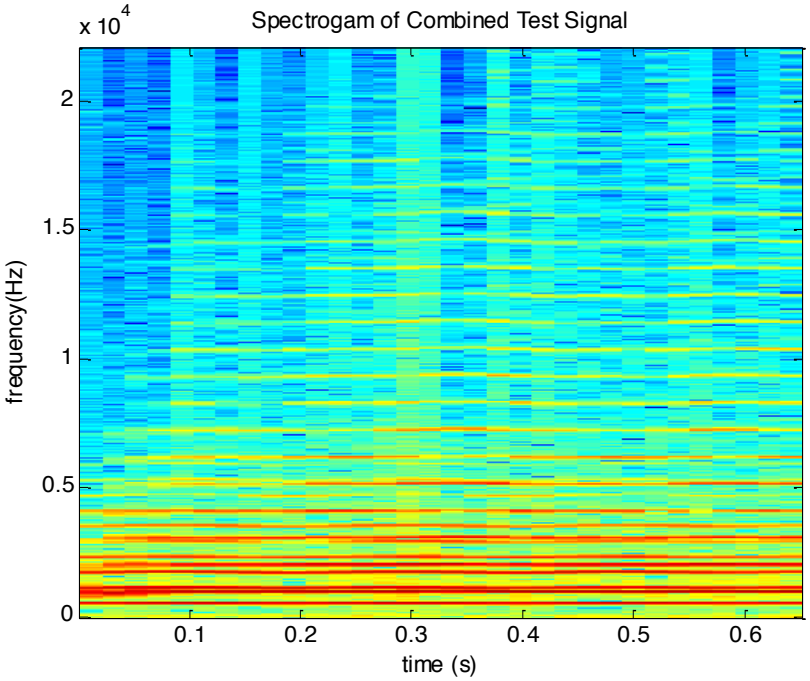

Figure 12. spectrogram of oboe note plus interference. Here we see the interferecne is weak but it significantly affected the performance.

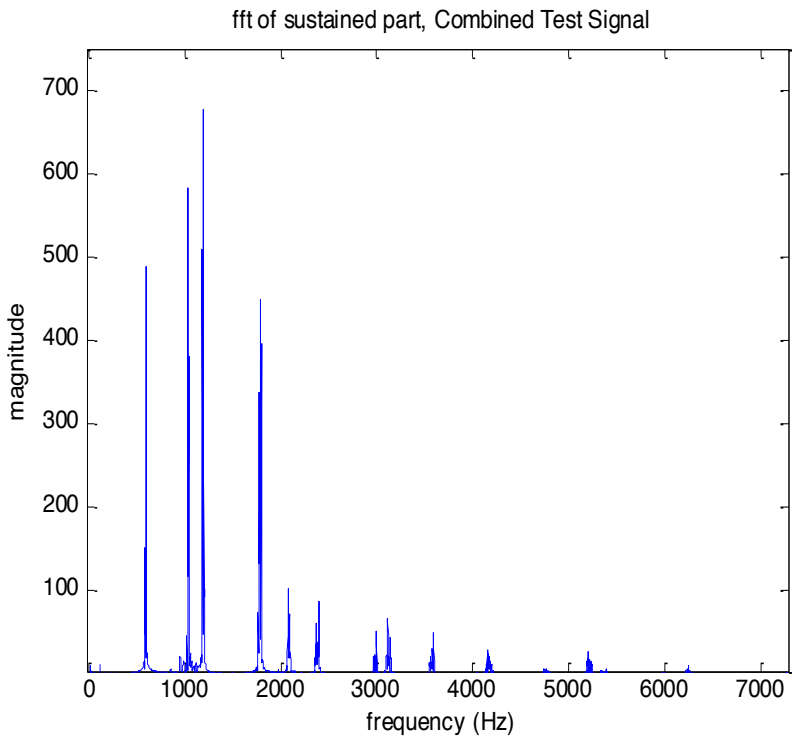

Figure 13. FFT of combined sound

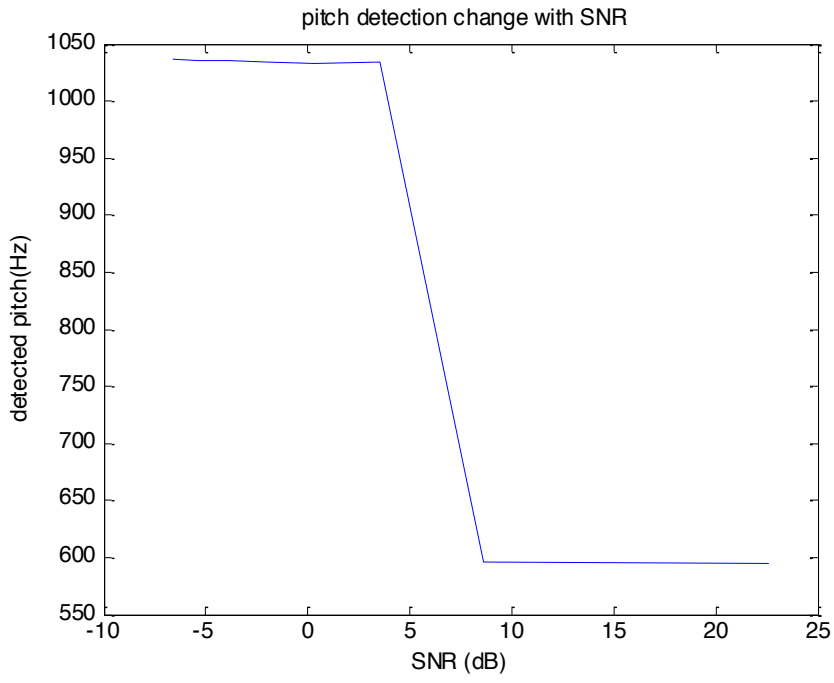

Figure 14. Detection result change with SNR.

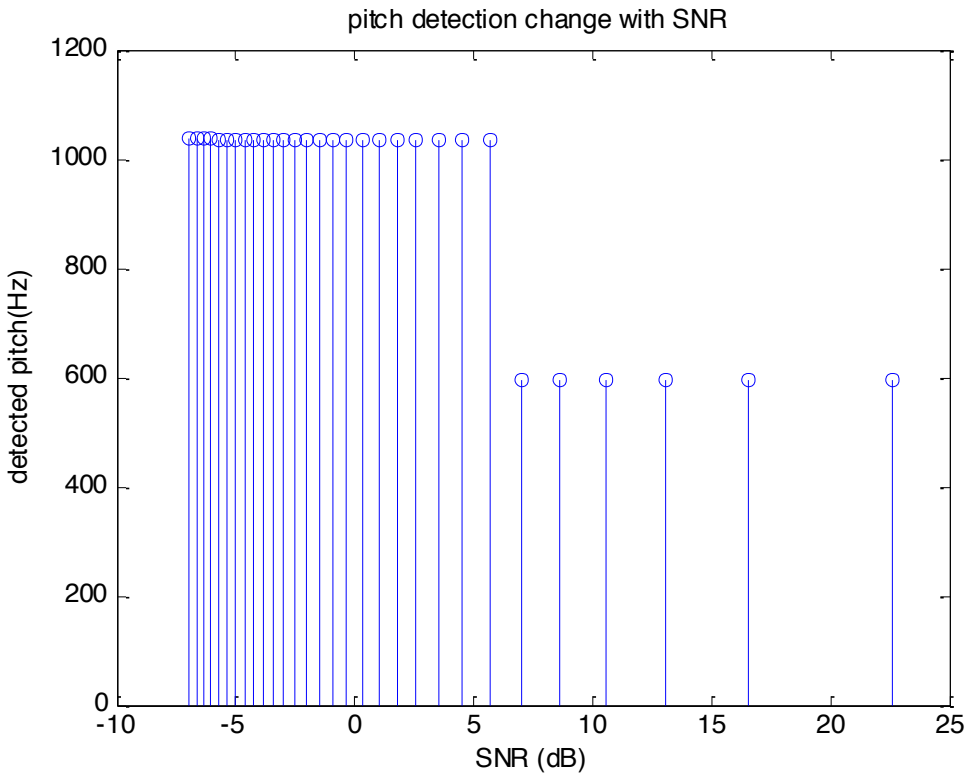

Figure 15. Detection result change with SNR, stem plot

3 Test factor 3: Noise

We then add in noise component here and test the detction performance. The source code for this part is provided at:PJ_CLASS_PITCHER_E03_Experiment_3.m and PJ_CLASS_PITCHER_E03_Experiment_4.m

The noise is much stronger tan concurring note in Sec. 3.1. Here MIR toolbox provide the right reading 596 Hz. The soectrogram is plotted in figure 16 and Figure 17. In Figure 19 we apply a lower SNR setting and still get good results (600Hz). From this experiemnt we see the algorithm is noise robust but not interferecne robust.

Further experiments are included in Figure 20 and Figure 21 (Source code: PJ_CLASS_PITCHER_E03_Experiment_4_morepoint.m). From here we see a jump at -16db. The other part is accurate. So in SNR we can conclude that MIR pitch detection is very robust for noise.

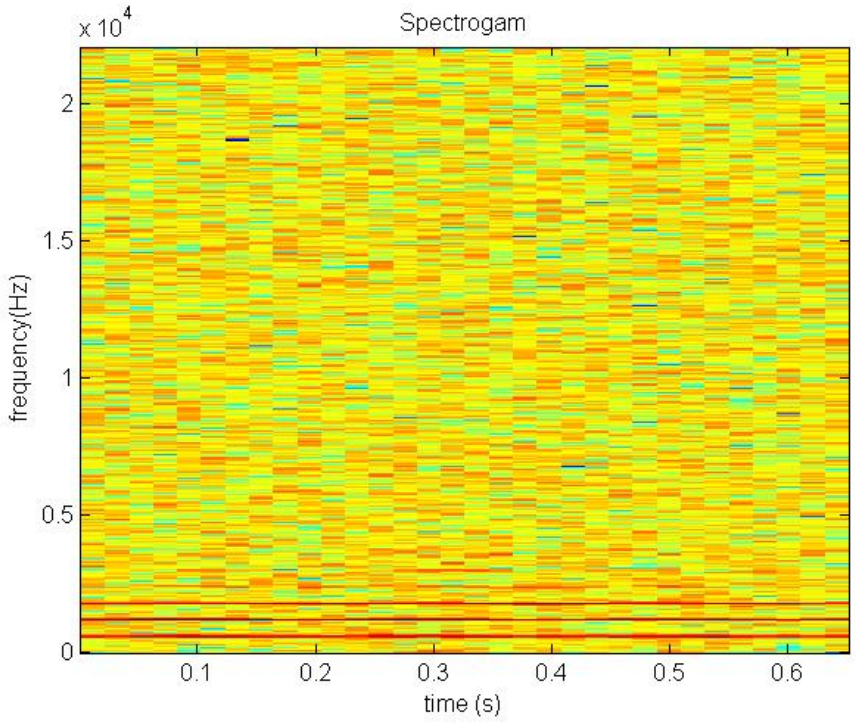

Figure 16: Experiment setting that adds in strong noise component. Here we can still get satisfactory pitch detection result.

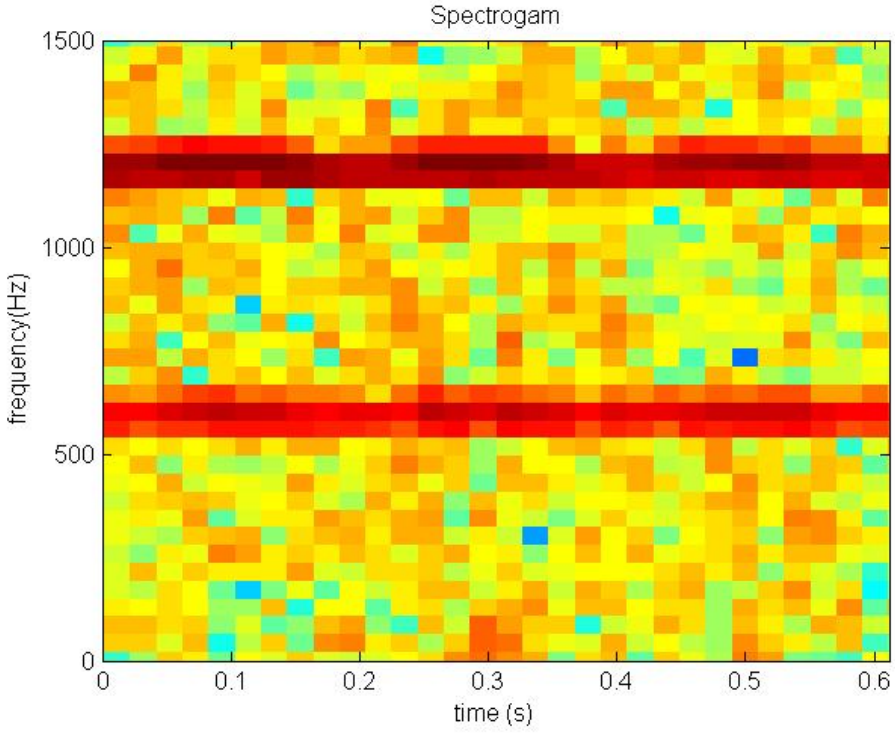

Figure 17: Zoom in of Figure 10 near actual pitch frequency (596Hz).

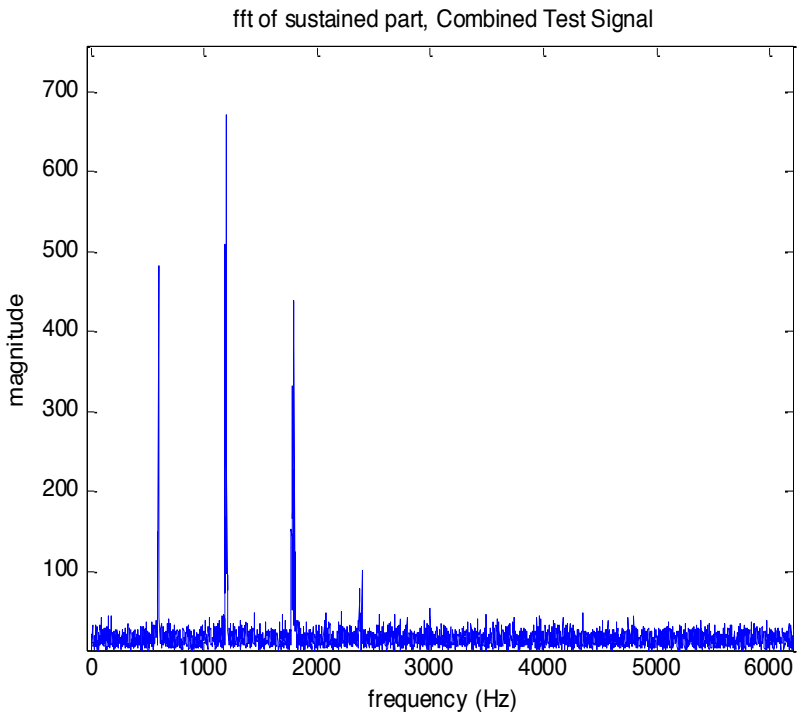

Figure 18: FFT of Experiment setting that adds in strong noise component. Here we can still get satisfactory pitch detection result in spite a high noise floor.

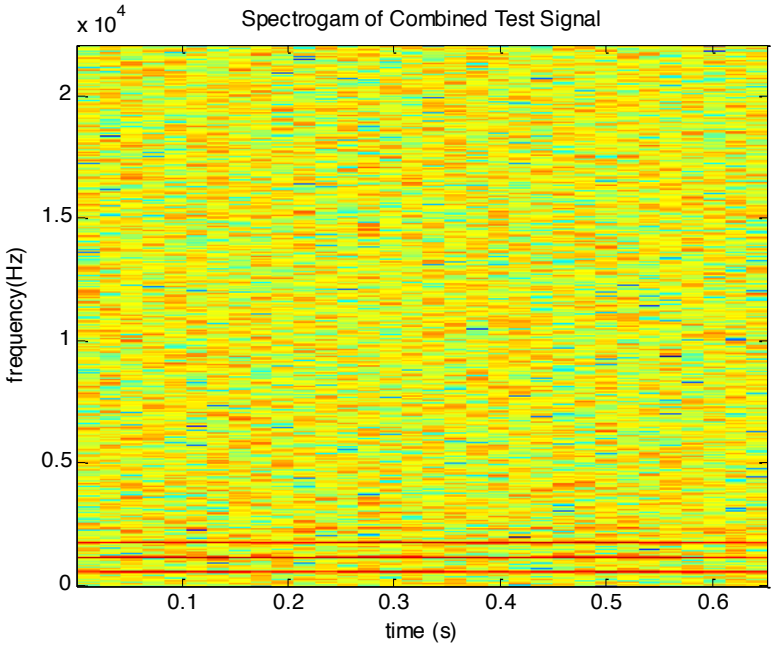

Figure 19. A lower SNR setting (-2.63) and we still get right result (600Hz).

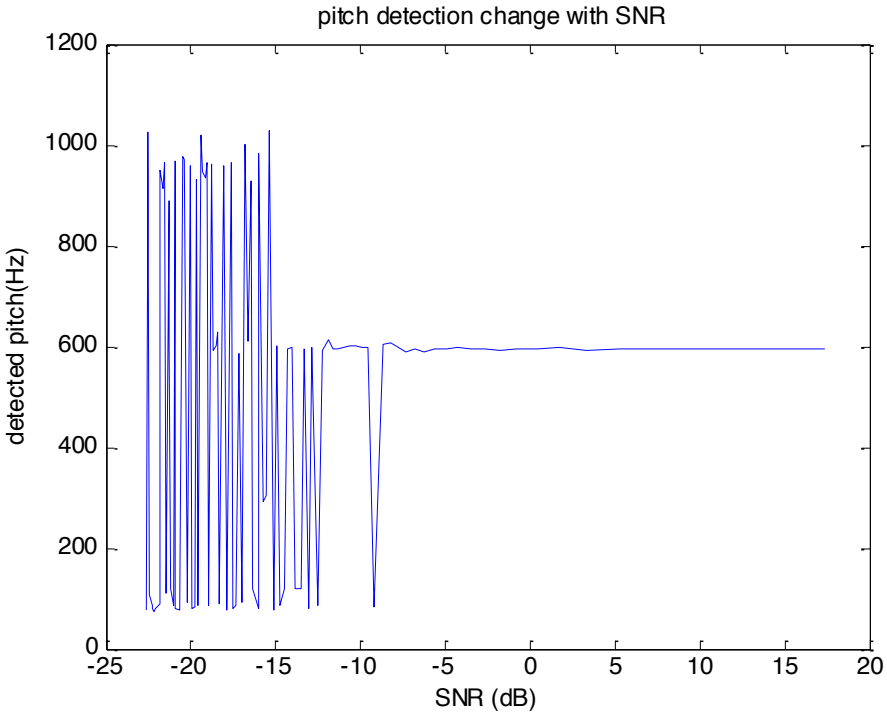

Figure 20. Detection result change with SNR.

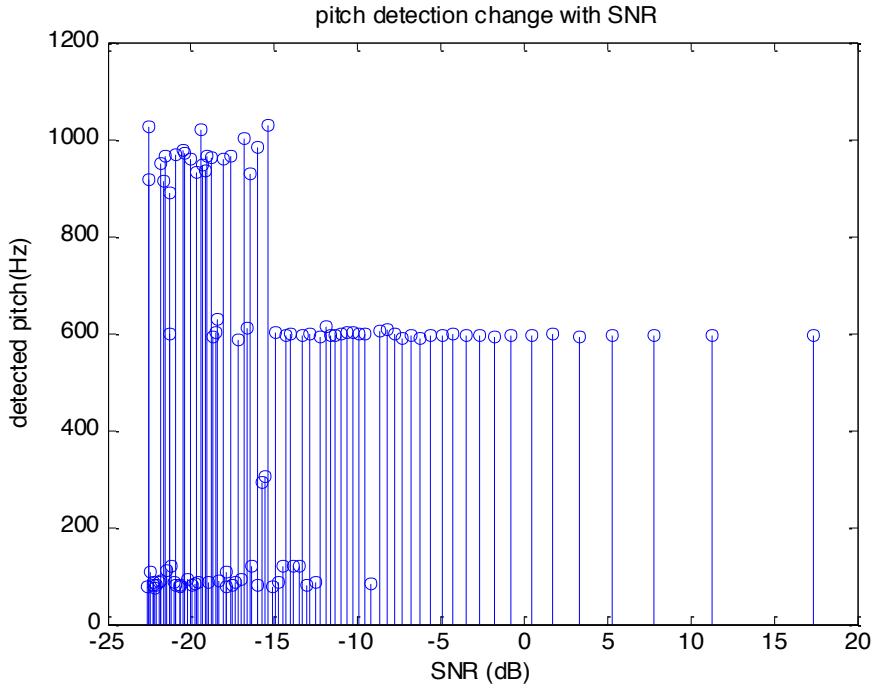

Figure 21. Detection result change with SNR, stem plot

5 Conclusions

We investigated several the possible cause of performance degradation of musical pitch detection algorithm. We elicit the degrading factors and will measure their effects in a simulated environment. We implement batch analysis of the influence of performance of these factors

- The different timbre of the same note, especially music note of different complexity
- Overlapping frequency domain contamination, such as incomplete polyphonic separation
- Concurring noise.

This study is useful for various applications. For example, in a record-and-search applications, a mobile phone user can record a song when he hear it at the supermarket. Then she can use the recording clip as a search query. In this case, the recoding is imperfect but we still want to get useful information.

References:

[1] MIR toolbox: Olivier Lartillot, Petri Toiviainen, "A Matlab Toolbox for Musical Feature Extraction From Audio", International Conference on Digital Audio Effects, Bordeaux, 2007.

[2] Musical sound sample: provided by Professor Mark Bocko and Professor Jame Beauchamp (UIUC)