Report

Problem 1-a

Code please see: myYin.m

If you need to run the code, you may evaluate codes in main.m, part 1-a.

Please note that I returned RMS value with the name “rms_val” instead of “rms”, because “rms” is a name of a Matlab function which might cause conflicts.

Here is an example of the processing on frame 1000:

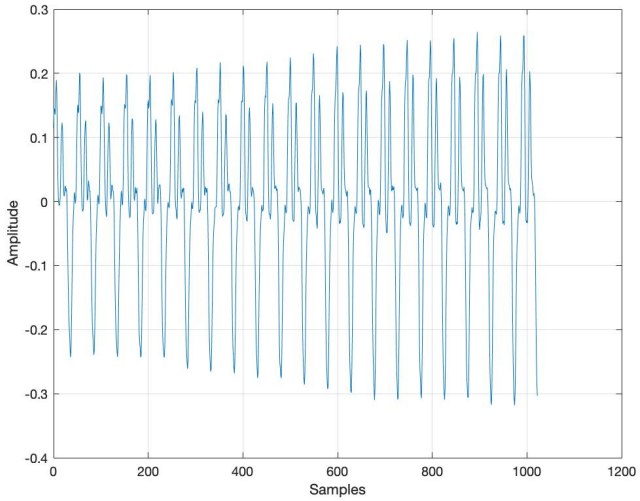

Figure 1 Waveform frame # 1000

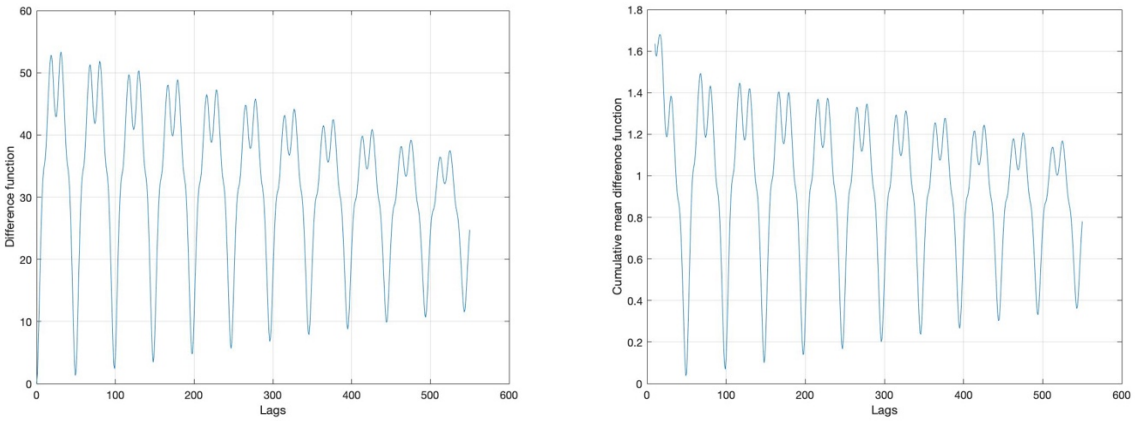

Figure 2 Difference function(left) and normalized one(right) frame #1000

Problem 1-b

Code please see: evalSinglePitch.m

If you need to run the code, you may evaluate codes in main.m, part 1-b.

Problem 1-c, 1-d

Code please see: main.m, part 1-c, 1-d

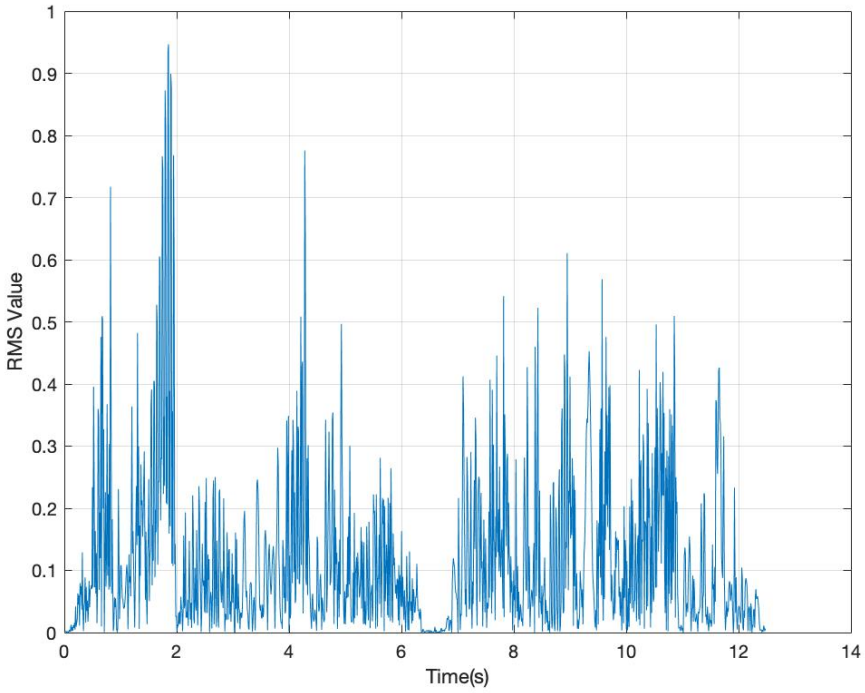

Figure 3 RMS value

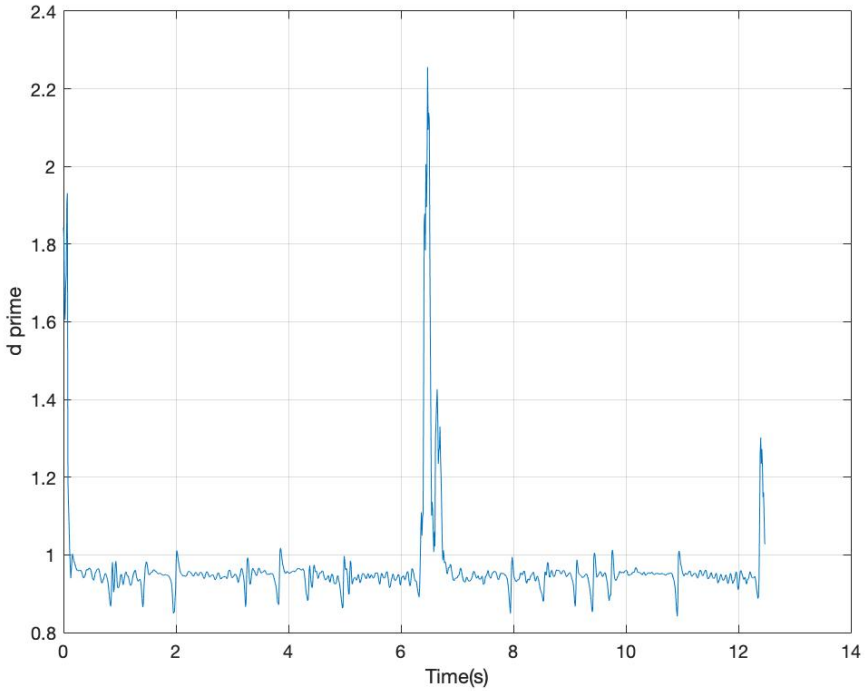

Figure 4 d_prime value

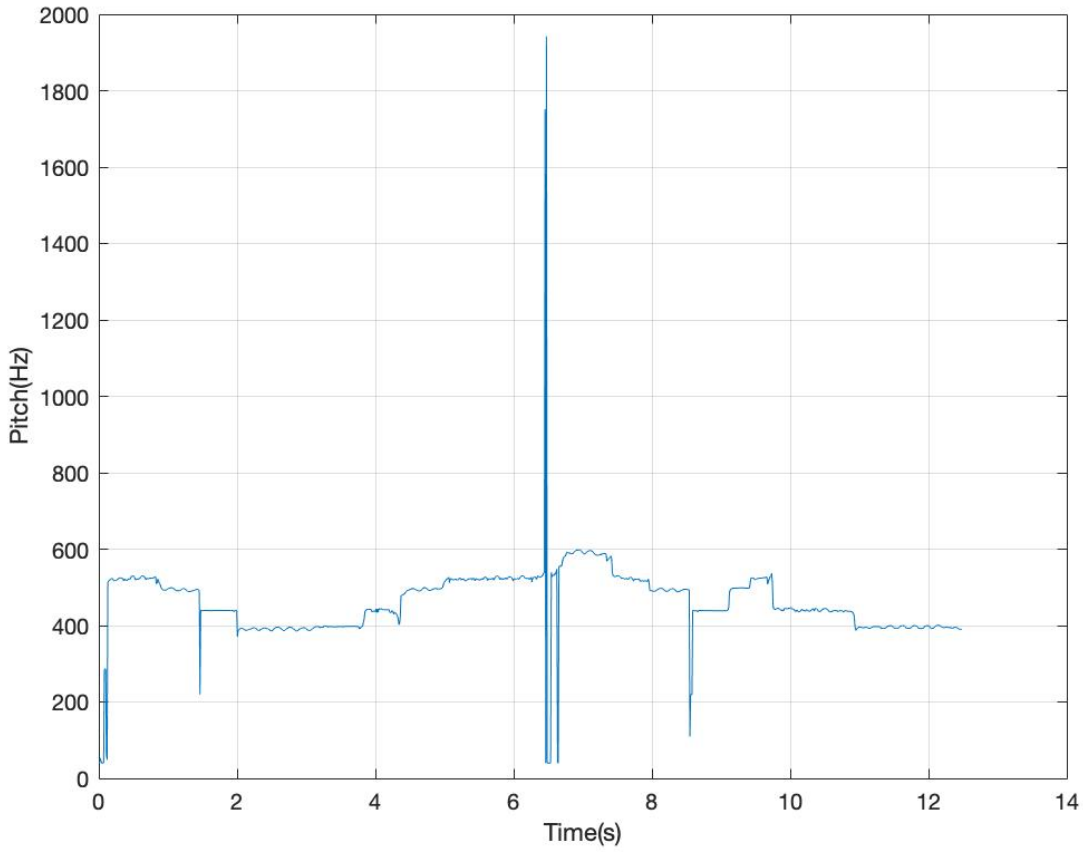

Figure 5 Pitch Estimation on violin.wav before manual correction

A grid search for the possible best RMS and ap_pwr thresholds is applied next. I searched the following values according to the observation of plots:

```
rms_thres = (0.001:0.001:0.1);
                        ap_pwr_thres = (0.6:0.05:1.2);
```

The grid search approved the result as follows:

	Original Estimation	Best Accuracy	Best F1-score
Accuracy	0.9705	0.9757	0.9705
Precision	0.8721	0.9288	0.8721
Recall	1.0534	0.9338	1.0534
F1-score	1.0534	1.0027	1.0534
RMS Threshold	-	0.9500	-
AP_PWR Threshold	-	0.01	-

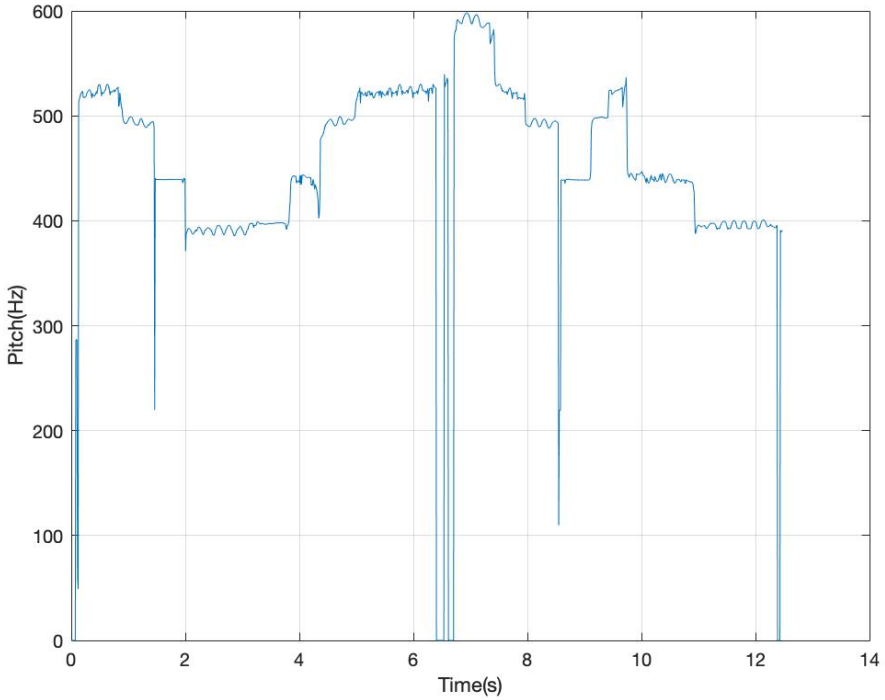

Figure 6 Pitch Estimation on violin.wav after correction

Problem 1-e

Codes: main.m, part 1-e

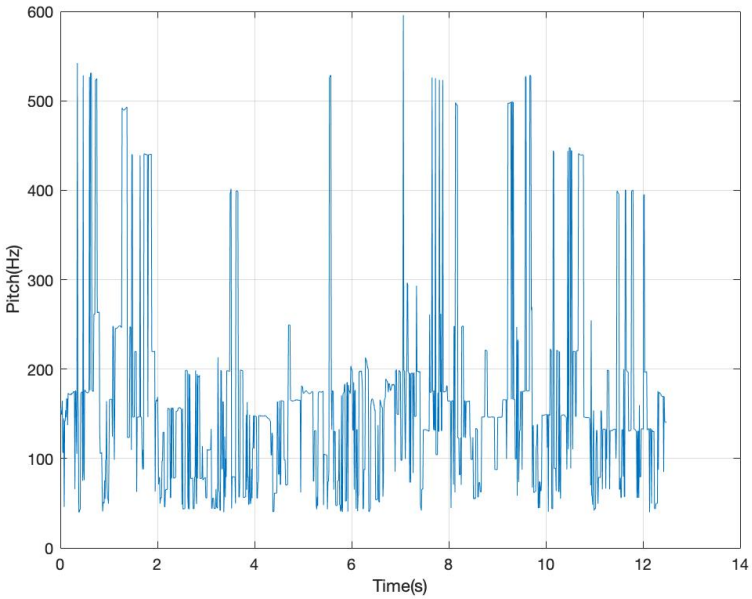

Figure 7 Pitch Estimation on violin_noise.wav

	Accuracy	Precision	Recall
violin.wav	0.9757	0.9288	0.9338
violin_noise.wav	0.1004	0.0904	0.0954
Change	-89.71%	-90.27%	-89.78%

Problem 1-f

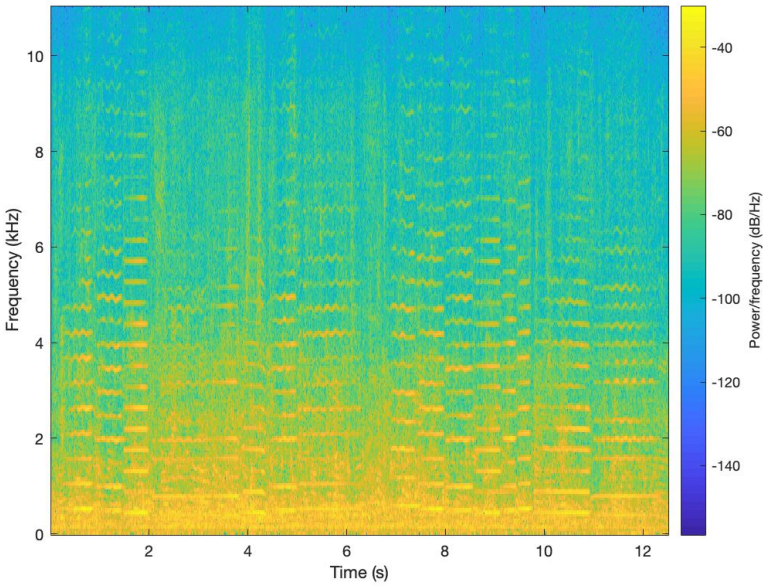

Figure 8 Spectrogram of violin_noise.wav

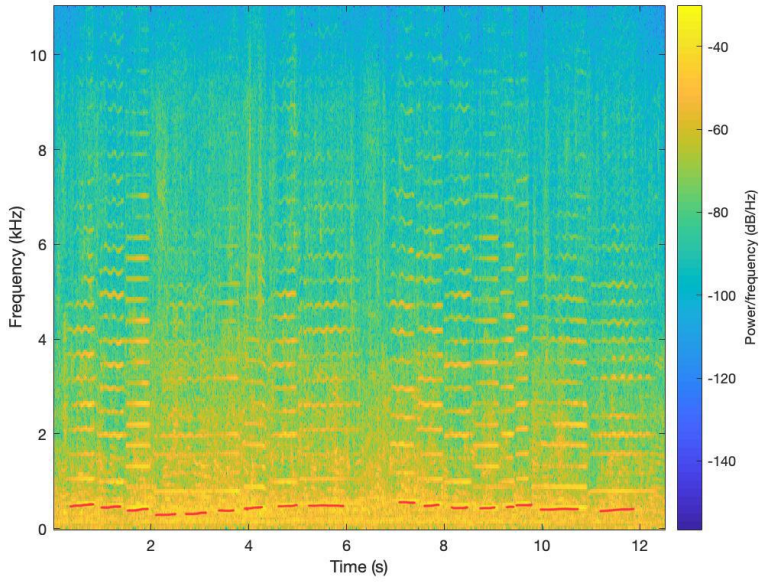

Figure 9 Spectrogram of violin_noise.wav with guessed pitch marked in red lines

My idea to better the pitch estimation of this file:

As it is indicated in the spectrogram above, the low-frequency part (1-1000Hz) is mixed with noise components, which makes it hard to distinguish instrumental sounds from the background noise. But when we turn to the higher frequency part, the instrumental sounds have very nice harmonics that are clearly detected. According to these characteristics, we may use methods based on spectrograms, find the high-frequency harmonics of the instrumental opponents and figure out their common divisor.

Also, timbre consistency might contribute to how humans separate the violin and noise sounds apart. We can detect timbre and consider its determines as pitch detection features.

Maybe we can improve the quality of the audio wave prior to the pitch estimation process. That means reducing noise affections, separating noise and needed soundwave apart before any training or pitch calculating. I read some literature in the speech enhancement filed because I think they have encountered many situations where the background babble noise needs to be removed. Spectral subtraction seems to be a promising idea as it is simple, classical and robust. The article is here: https://ieeexplore.ieee.org/document/1163209

Or, we can try deeplearning-based method. I've read some papers too and found the following one seems good:
CREPE: A Convolutional Representation for Pitch Estimation
https://pdfs.semanticscholar.org/86ae/ec4d48d949190b3a0c2bf32c101fc23f13a3.pdf

Problem 2-a

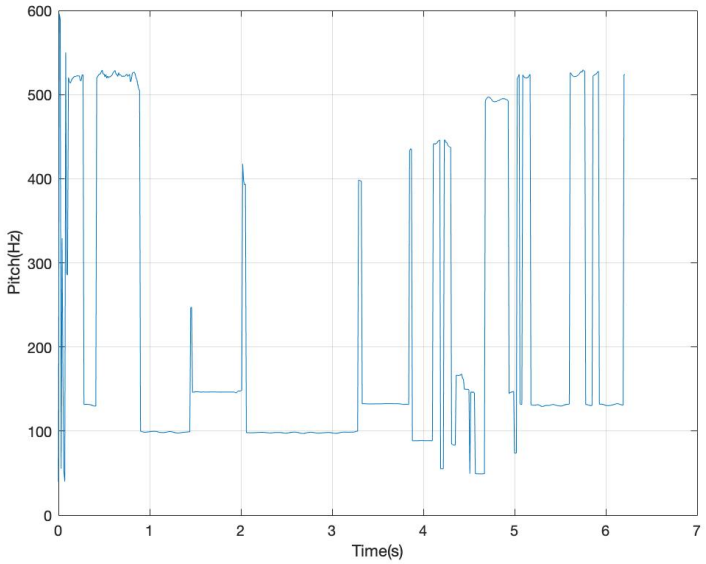

Figure 9 Pitch Estimation on violin_bassoon.wav

tHop=0.01; tW=0.0464; f0Min=40; f0Max=600; dp_th=0.1;

According to the wave file, the pitch goes up steadily and goes back. But we can see from figure 9, the pitch contour is not reasonable as the pitch jumps between higher frequencies to lower ones reciprocally. Moreover, most pitch estimations stayed in the lower frequency area, which means the algorithm was mostly detecting bassoon pitches.

Because the violin and bassoon sound in the dataset have very different pitch ranges, so I tried using higher frequency limits parameters to detect violin and lower ones to detect bassoon. The results are as figure 10 and 11:

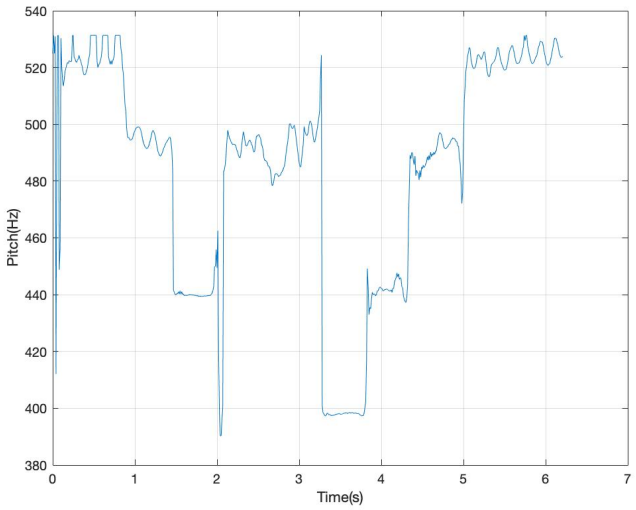

Figure 10 Violin Pitch Estimation on violin_bassoon.wav

tHop=0.01; tW=0.0464; f0Min=390; f0Max=530; dp_th=0.1;

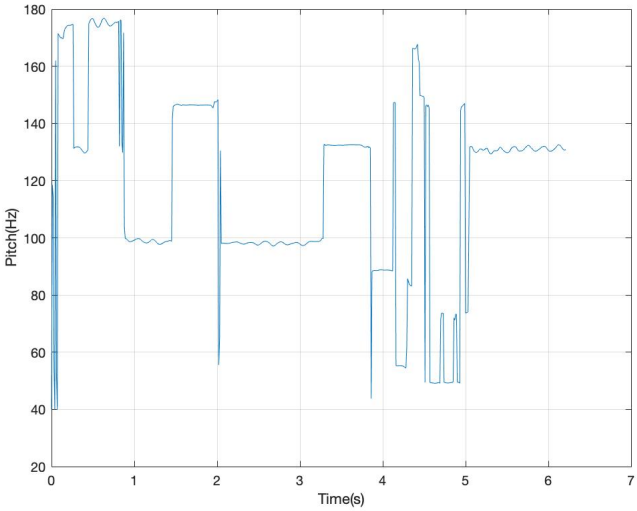

Figure 11 Violin Pitch Estimation on violin_bassoon.wav

tHop=0.01; tW=0.0464; f0Min=40; f0Max=180; dp_th=0.1;

There are many problems. For example, in figure 10, time 2s - 3s, the frequency is about 2 steps higher than it should be, and weirdly the frequency in the following 3.2s-3.8s is back to normal. Changing the absolute threshold did not fix the problem and these two periods have almost the same energy.

Problem 2-b

I modified myYin.m to output multiple pitches at a time, code in myYinPoly.m. But it did not work well, just as the homework introduction suggested: the two pitches are either an octave apart or about the same heights. The first issue is due to some harmonics of one instrument are still deeper than the "digs" of the other instrument, so the algorithm chooses one and its harmonic. The second problem is because sometimes there is no dig below the threshold and the algorism picks two global mins that are close in value. I did not find a good way to fix them yet.

Problem 2-c

Apart from using continuity to connect pitches into pitch contours, I think we can also consider composition rules and conventions, or sound characteristics of different instruments, also genre variations, as we are dealing with music pieces.

For example, different parts of music may have the different possibility of abrupt or large changes in pitches, Largo parts pitches may change more softly than Allegro parts. Pitches of percussive may change more suddenly than that of string instrumentals. We can design thresholds accordingly or track the ideal pitch contours and see if the estimated pitch contours follow it.

The natural language processing filed might encounter similar problems as they need to predict what the next word or sentence is according to the context consistency. We can use their methods to predict the following pitches and attach the most possible real F0s to the same line. Or we can see this as a classification or prediction problem and train a recursive neural network to do the job.