EQ2340 Pattern Recognition Project Assignment 2 Report

Feiyang Liu (liuf@kth.se)
Baoqing She (baoqing@kth.se)

***Abstract*—HMM[1](Hidden Markov Model) is widely used to characterize signals in many fields nowadays, due to its superiority, such as simple mathematical structure, describing complex feature-vector sequences, automatically adapted among others. We want to create and analyze HMM in our project.**

I. Introduction

In this report, We are supposed to design a feature extractor for a Song Recognition system capable of recognizing short hummed, played, or sung melodies. In practice, the input often includes too much data and we need to extract the signal aspects we are interested in to help remove redundancy for HMM. All of our simulations are based on MATLAB, and all relevant source codes are attached in zip archive.

II. Theoretical Knowledge

To design a feature extractor for song recognition, we are supposed to know a little about music signals, Fourier analysis, music theory, and human pitch and loudness perception. For brevity, we will not include too many basic knowledge. For more details, refer to [1].

We extract data of melody using Fourier analysis, which implements Fourier transform in many short pieces using window frames. However, there too many data after Fourier analysis. Using the given function $[frIsequence] = GetMusicFeatures(signal, f_s)$, we choose three kinds of melody features in this task i.e. estimated pitch f_t, estimated correlation coefficient r_t between adjacent pitch periods and frame root-mean-square intensity I_t. There still pitches corresponding to silent pauses in the sequence. In next section, we will talk about how to set threshold to filter out them.

III. feature extractor design

Feature matrix $frIsequence$ are given in (1).(where T is the number of analysis frames, which depends on the signal duration. For each frame t the matrix contains an estimated pitch f_t in Hz, and estimated correlation coefficient r_t between adjacent pitch periods, and the frame root-mean-square intensity I_t.)

$$frIsequence = \begin{bmatrix} f_1 & f_2 & ... & f_T \\ r_1 & r_2 & ... & r_T \\ I_1 & I_2 & ... & I_T \end{bmatrix} \quad (1)$$

Fig.1 gives frequency pitch of three melodies. We place time along the horizontal axis and set the vertical axes to logarithmic scales. To see the melodies more clearly in the pitch profile plots, we look at the frequency range $100 - 300$ Hz especially. The shape of pitch frequency of melody1 and melody2 are similar, while the frequency fluctuation of melody3 differs. Since in Fig.1 frequency pitch also includes

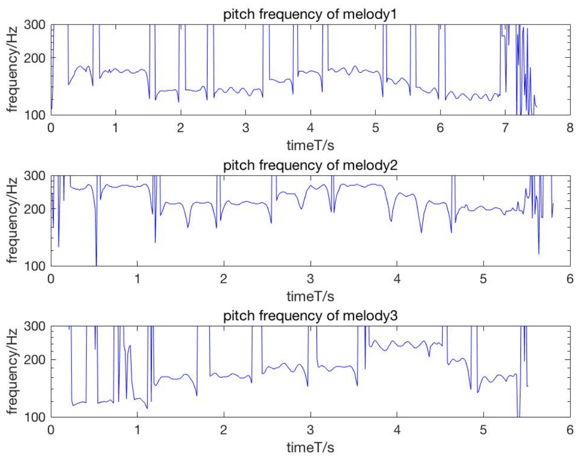

Fig. 1: Estimated frequency pitch of three melodies

the time duration dominant by background noise, our feature extractor is designed to delete those snippets.

Step 1. Eliminate the Non-harmonic Components in the Pitch

1) *Using Correlation Finding Pause:* In Fig.2, the estimated correlation coefficients between pitch periods of different melodies are demonstrated. As we can see, at the end and beginning of the melody, correlation coefficients are low, since there are silent pauses. Hence, For small correlation coefficient, we consider it is where pause locates.

To design a flexible threshold for pause finding, we assume when the correlation coefficient is large than 0.95, there is no pause in the time duration. Then, we compute the average value of coefficients smaller than 0.95 to get the threshold T_{sigma}, which is the red dash horizontal line in Fig.2. The correlation coefficients below the red line are caused by silent pauses.

2) *Using Normalized intensity Finding Pause:* In Fig.3, the intensity of melodies are plotted. The shapes of the intensity are similar for melody1 and melody2, since they are from the same songs. Since for different signals the intensity can

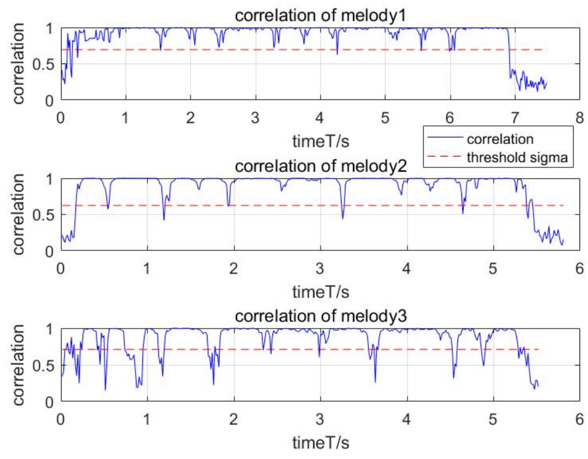

Fig. 2: Estimated correlation coefficient of three melodies

be arbitrary value, we normalize the intensity to set standard threshold for intensity.

In the same manner of correlation method, we assume when the intensity is large than 0.8, no pause in the time pieces. Then, the average value for normalized intensity no larger than 0.8 is computed. The mean value is the threshold T_{theta} for normalized intensity, which is the red dash line in Fig.4. The normalized intensity under the threshold is considered as intensity of noise, and the corresponding time is where the pause locates.

Based on Fig.2 and Fig.4, use the threshold to first find the non-harmonic components in the pitch, set them as low-bound (the minimum value of frequency pitch). Since noisy pauses have much higher frequency compared with notes signals. In this project, we assume frequency large than 500 Hz is the frequency belonging to noise and we set them to low-bound as well. Then we get a new pitch sequence S_1 with almost harmonic regions kept.

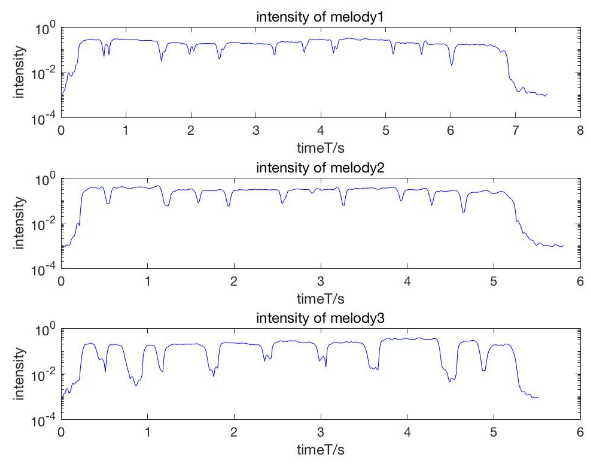

Fig. 3: Estimated intensity of three melodies

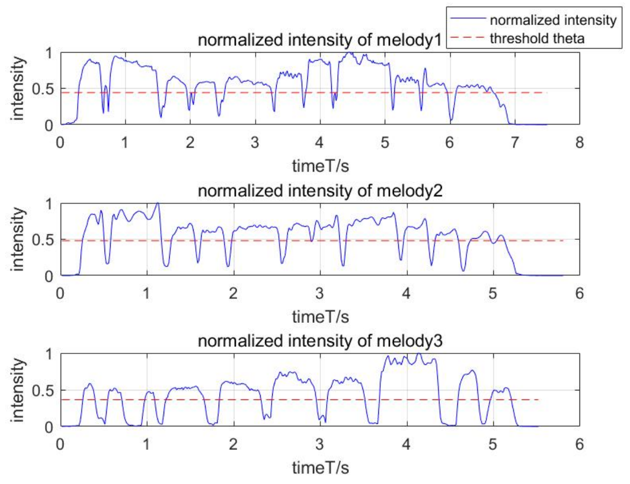

Fig. 4: Estimated normalized intensity of three melodies

Step 2. Robust to Transposition

Using the sequence S_1 obtained in last subsection, first logarithm it. Then, find the minimum frequency except for low-bound and extract it from the whole sequence to get new sequence S_2, which ensures that this algorithm is robust to transposition.

Step 3. Add Noise to Low-bound

Since noises also carry sufficient information, here we add noises to regions where we label them as non-harmonic snippets i.e. low-bound in the previous steps, instead of deleting them directly. Here noises are generated from normal distribution, where the frequency of harmonic regions give the mean and standard deviation. Then we get the new sequence S_3 which takes noises into consideration as well.

Step 4. Frequency to Semitones

In this step, we transform the pitch feature sequence S_3 obtaining in the previous step into semitones feature sequence S_4. Results are shown in Fig.5. An octave is a big leap on any frequency scale which is further subdivided into twelve smaller steps, known as semitones. For brevity, we will not give details about octave and semitones. More information refers to [1].

Since the lowest octave is $20Hz - 40Hz$, and in logarithmic scale it has 12 equal length semitones, which indicates that the length of every semitone is equal to $(log(40) - log(20))/12$. Using this knowledge and the equation, we divide the previous sequence by $log(2)/12$, the returned sequence is how many semitones are there between this frequency and the lowest frequency. And this is also our output feature.

As we can see from Fig.5, the output feature of melody1 and melody2 are almost identical with the same upperbound and lowerbound, while differs from the feature of melody3.

IV. VERIFY OUTPUT FEATURE OF FEATURE EXTRACTOR

To verify that our features are transposition independent, multiply the pitch track in matrix $frIsequence$ by 1.5, and

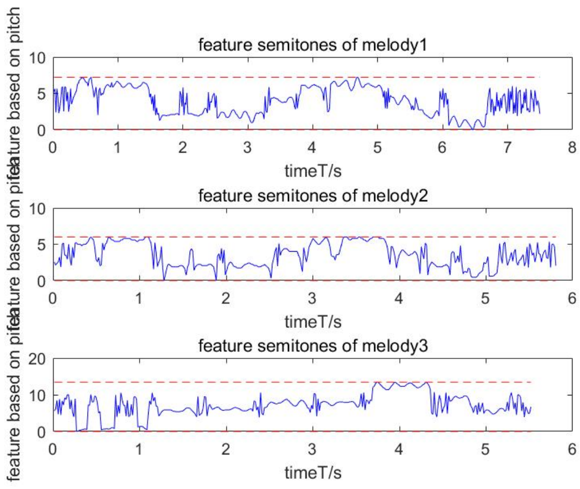

Fig. 5: Output feature based on semitones of three melodies

use this in our feature extractor. The output are shown in Fig.6. the result is almost virtually the same as with the original pitch. There is still some differences between two outputs, the reason is that noises are given randomly in non-harmonic regions.

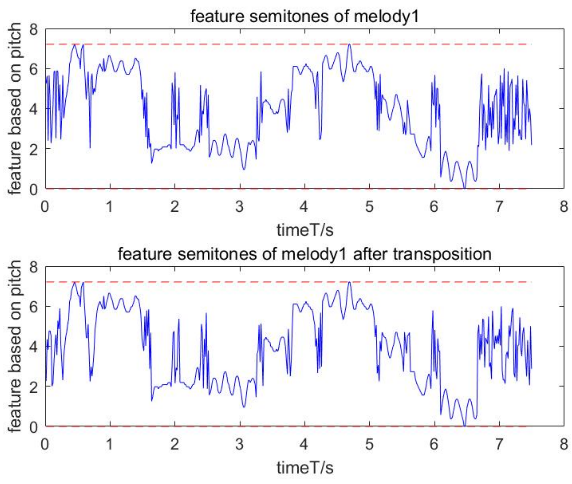

Fig. 6: Output feature based on semitones of melody1 and its transposition

V. Discussion

In this section, we think about the cases where this proposed feature extractor may perform poorly.

Problems may happen when the original song volume is low or changes rapidly, then intensity variation between melody and silent/noise is not apparent. Therefore, using threshold will delete non-harmonic region as well as harmonic melody snippets.

Hence, it is able to have two different output feature from one same song. To be sepecific, using one same song, the first recorded melody is played in the normal volume while keeping changing the volume of the other one. Under this circumantance, outcome feature should be simiar. However using our feature extractor, some of harmonic snippets will be repplaced by noises for the second record. This is because snnipets with low volume is eliminated by a higher threshold.

As for the opposite situation, from my perspective, this feature extractor is not able to generate simiar outcome feature sequences from two totally different records.

References

[1] Arne Leigon and Gustav Eje Henter. Pattern Recognition Fundamental Theory and Exercise Problems