EQ2341 Pattern Recognition and Machine Learning Assignment 2

Chenting Zhang
chzha@kth.se

Hang Qin
hangq@kth.se

May 9, 2023

1 Introduction

In this project, we have designed a "query by humming" program which aims to distinguish between a few brief snippets of whistled or hummed melodies. Firstly, we analyzed the pitch, correlation and intensity information of the given three hummed melodies. After that, we compared the feature of each melody based on our feature extractor. In addition, we proved that our feature extractor is robust and thus can pass through several variation tests. Finally, we illustrated the limitation of our model and feature extractor design.

2 Song file detail

The song files are three *.wav* files collected from the same man's humming with lengths from 5 to 7 seconds. The first song and the second song are the same melody but in different octaves, and the second one is slightly faster. The third song sounds like a different melody.

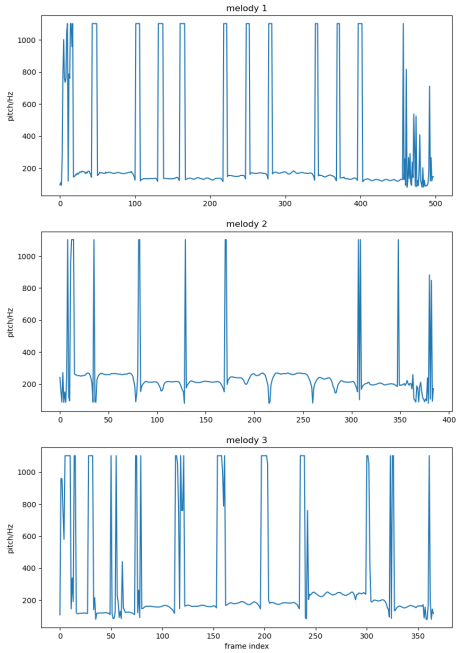

Figure 1: Pitch profile of the three song files

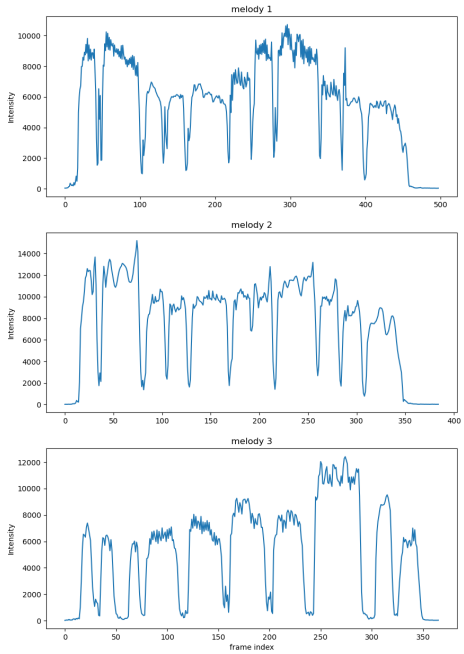

Figure 2: Intensity profile of the three song files

Figure 1 and Figure 2 show the pitch and intensity profiles of the three recordings respectively. One can tell from these plots that melody 1 and melody 2 are more likely to be the same melody because their intensity and pitch profiles are more similar.

3 Feature extractor design

After analyzing the data of the melodies, we designed a feature extractor. Due to the basic music theory, music is boiled down to its bare essentials: the notes, the succession of notes played, their durations, and the durations of pauses between them as well as the sound volume. We partitioned the extractor model into two parts. Firstly, we detected how many notes are played in this melody from the pitch, correlation as well as intensity information, secondly, we wrote a program of roughly recognizing which semitones it represents and which octave it belongs to.

3.1 Note detection

The silence mainly consists of two components. One is the silenced frames at the beginning and the ending of the *.wav* files which have low normalized intensity values and thus can be filtered out by setting a lower bound. The other kind of silenced frames is brief breaks between successive voiced notes. In this case, the sound intensity can still be fairly high due to the impact of environmental noise. So correlation coefficient between pitch periods should also be considered. Correlation in signal segments with a clear pitch could be high while much lower in noisy regions. By using this property, we could detect the silenced region between notes. Besides, we also filtered out the abnormal high frequency part, we only retain the pitch in the frequency range between 32.703HZ and 1046.502HZ which is the standard frequency of C1 and C6.

3.2 Note recognition

After that, we filtered out the voiced segment based on the zeros in the sequence. We used the mean value of the specific voiced frames as the representation of that note.

Then we need to recognize which note it played in a typical keyboard setting. Based on the music theory, the C note frequency from C1 to C7 are [32.703, 65.406, 130.813, 261.626, 523.251, 1046.502, 2093.005]Hz. Plus, there are 12 semitones in one octave and the quotient between the pitches of two adjacent semitones is the same everywhere. Thus, we firstly transfer the pitch in logarithmic scale and then divided it into 12 parts. We calculated the relative semitones and the octave it belongs to, subsequently the absolute semitones. We also marked each voiced frames with the corresponding notation"C", "C#", "D", "D#, "E", "F", " F#", "G", "G#", "A", "A#", "B". Then we normalized the absolute semitones into the (0, 120) region as positive integer.

$$\text{Feature}[i] = \lfloor \frac{\text{semitone}[i] - \min[\text{semitones}]}{\max[\text{semitones}] - \min[\text{semitones}]} \times 120 \rfloor$$

For example, for the first voiced region in melody1 with the frequency 168.4356, it belongs to the third octave region with relative semitone 4, so it belongs to E note. This discrete vector is the final feature we obtain.

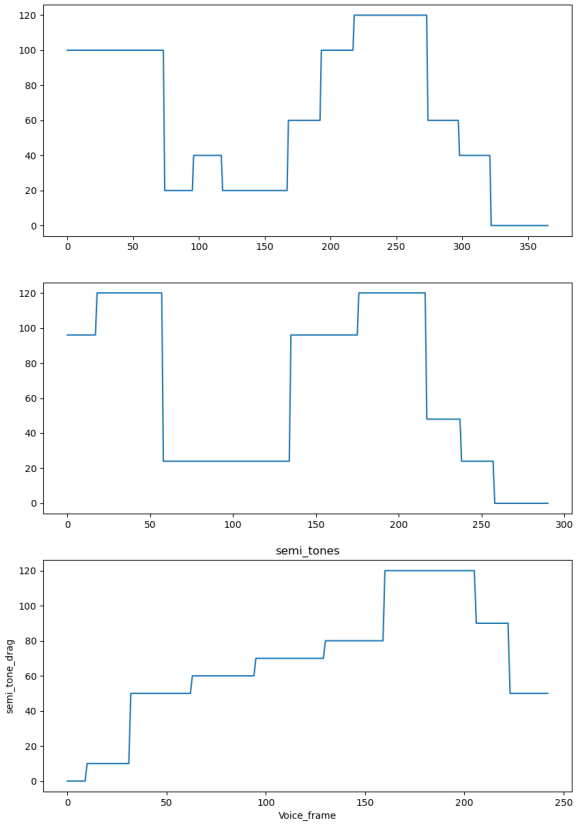

Figure 3: Note detection

We could tell from figure6 that melody1 and melody2 have similar features, while melody3 is completely different.

4 Verification

To verify our feature extractor is robust, we test it in three cases to check whether it is transposition invariant, volume invariant and octave jump invariant.

4.1 Transposition invariant

To verify that features are transposition independent, we multiply the pitch track returned by GetMusicFeatures by 1.5 and use the feature extractor to process it. Figure 4 shows that the output is virtually the same as the original pitch.

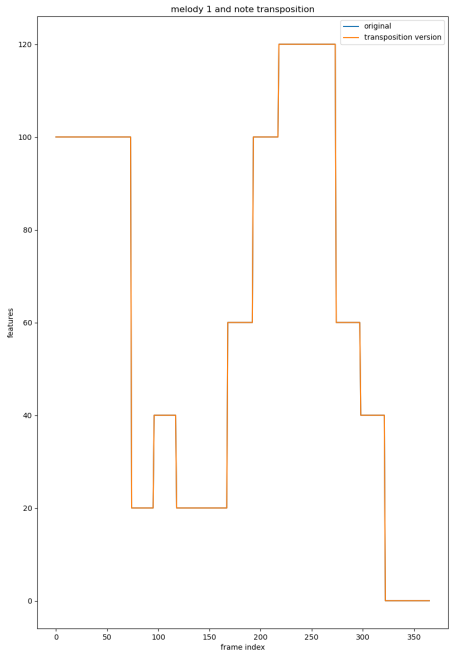

Figure 4: Comparison of features before and after pitch transposition

4.2 Volume invariant

To verify that features are volume invariant, we multiply the original waveform by 1.5 and use the feature extractor to process it. Figure 5 shows that the output feature is virtually the same as the original one.

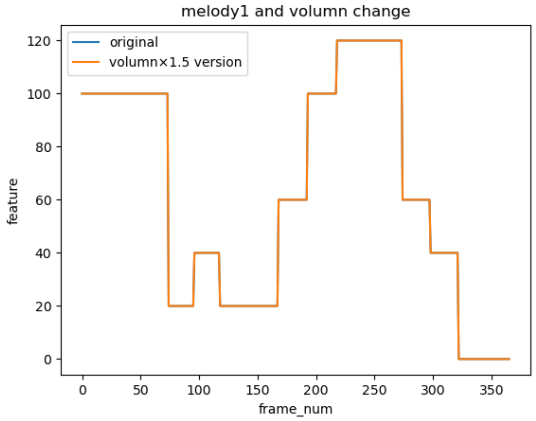

Figure 5: Comparison of features before and after volume changes

4.3 Octave jump invariant

To ensure that our feature extractor would not be overly sensitive to brief episodes where the estimated pitch jumps an octave (the frequency suddenly doubles or halves), we design an algorithm that can eliminate short sudden changes in the feature sequence that are less than the set length. Then, we choose 5 random time steps and double the pitch here. Figure 6 shows the feature without using our octave jump invariant algorithm. But when our algorithm is used, these sudden changes will not affect the feature and the result is exactly the same as presented in Figure 3.

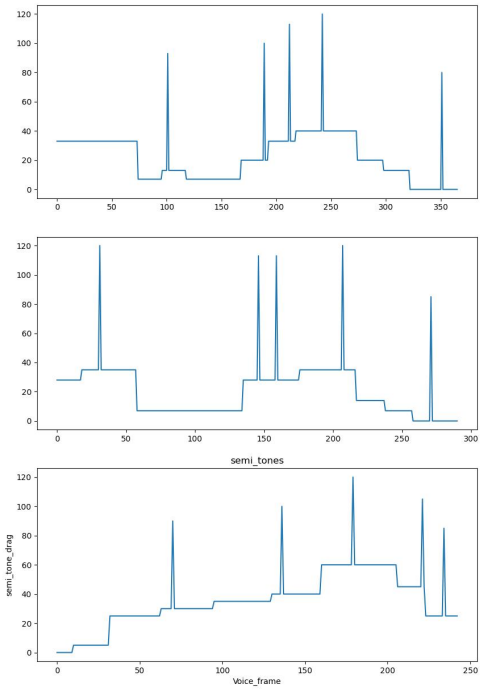

Figure 6: Features without octave jump invariant algorithm

5 Limitation

The designed feature extractor has some limitations:

1. This model can only be used to deal with one note. Due to the fact that the provided GetMusicFeatures function cannot handle polyphonic sounds, several notes played simultaneously cannot work. For example, chord detection is not feasible.

2. It cannot address tempo issues, meaning that two identical melodies with faster and slower tempos cannot be detected as the same piece. This is due to the fact that we only filtered out the silenced part but do not address the problem of notes with different durations.

3. Human voice can never be as precise as machine-generated sound, meaning that perceptually similar notes may be classified as different notes. This issue reveals in the feature of melody1 where the second note should be C constantly while a deviation C# has occurred in the middle for a short time.