CMPE362

Project

Ahmet Fırat Gamsız – 2020400180

Fatih Demir – 2020400???

**Algorithm 2**

**Frequency selected processing combfilters algorithm #1**

Beat Detection Algorithm Report

1. Code Explanation

a. filterbank

The filterbank function divides a time-domain signal into individual frequency bands. Here's a breakdown:

Inputs and Defaults: The function accepts a signal sig, band limits bandlimits, and a maximum frequency maxfreq. Default values are [0 200 400 800 1600 3200] for bandlimits and 4096 for maxfreq.

Fourier Transform: The signal is converted from the time domain to the frequency domain using FFT.

Band Limits Calculation: The function converts band limits from Hz to the corresponding points in the FFT output.

Band Creation: Portions of the FFT output are allocated to different frequency bands.

Plotting: The function visualizes the frequency bands.

b. hwindow

The hwindow function rectifies a signal and convolves it with a half Hanning window. Here's its workflow:

Inputs and Defaults: It accepts the signal sig, window length winlength, band limits bandlimits, and maximum frequency maxfreq. Defaults are 0.4 seconds for winlength, [0 200 400 800 1600 3200] for bandlimits, and 4096 for maxfreq.

Half-Hanning Window Creation: A half-Hanning window is created.

Inverse FFT: The frequency domain signal is transformed back to the time domain.

Rectification and Re-FFT: Full-wave rectification is performed in the time domain, and the signal is transformed back to the frequency domain.

Convolution: The signal is convolved with the half-Hanning window by multiplying their FFTs.

Plotting: The resulting signal is plotted.

c. diffrect

The diffrect function differentiates a signal and performs half-wave rectification:

Inputs and Defaults: It accepts a signal sig and the number of frequency bands nbands, with a default of 6.

Differentiation and Rectification: The signal is differentiated and then half-wave rectified.

Plotting: The function plots the differentiated and rectified signal.

d. timecomb

The timecomb function determines the tempo of a musical signal divided into frequency bands:

Inputs and Defaults: It takes a signal sig, beat resolution acc, minimum BPM minbpm, maximum BPM maxbpm, band limits bandlimits, and maximum frequency maxfreq. Defaults are 1 for acc, 60 for minbpm, 240 for maxbpm, [0 200 400 800 1600 3200] for bandlimits, and 4096 for maxfreq.

Comb Filter Application: A comb filter is applied to identify the BPM by analyzing the signal's periodicity.

BPM Calculation: The BPM is determined by finding the frequency with the highest energy after applying the comb filter.

Plotting: The output signal is plotted.

e. freqselectedcombfilter

The freqselectedcombfilter function predicts the BPM of a given song:

Inputs and Defaults: The function takes the name of a .wav file song and maximum frequency maxfreq, defaulting to 44100.

Sample Extraction: The middle 5 seconds of the audio file are extracted.

Band Limits Creation: Band limits are created logarithmically.

Beat Detection: The beat detection process involves:

Filtering the signal into frequency bands using filterbank.

Applying a half-Hanning window with hwindow.

Differentiating and rectifying the signal using diffrect.

Applying comb filters recursively with timecomb to refine the BPM estimate.

BPM Output: The final BPM is output.

2. Results

To evaluate the algorithms, we applied them to various songs. The BPM detection process involves:

Filtering: Decomposing the signal into frequency bands.

Windowing: Applying a half-Hanning window.

Rectification: Differentiating and half-wave rectifying the signal.

Comb Filtering: Using a comb filter to detect periodicity and estimate the BPM.

Song 1: Example Song 1

Predicted BPM: 120

Process: The signal was decomposed, windowed, rectified, and comb-filtered to identify a BPM of 120.

Song 2: Example Song 2

Predicted BPM: 95

Process: Similar steps were followed, yielding a BPM of 95.

Comparison

Accuracy: The algorithm performs well for steady beats and simple rhythms. More complex or varying rhythms might require further refinement.

Song Type: The accuracy is generally higher for songs with a clear, regular beat.

3. Conclusion

The beat detection algorithms implemented show promising results for BPM detection in various songs. However, some areas need improvement:

Shortcomings:

Accuracy decreases with complex rhythms or varying tempos.

Computational efficiency can be improved.

Potential Bugs:

The recursive calls in timecomb might lead to inaccuracies if not carefully managed.

Edge cases with extremely high or low BPMs might need handling.

Ideas for Improvement:

Implement adaptive windowing to handle varying tempos.

Optimize the comb filter application for better performance.

Incorporate machine learning techniques to enhance BPM prediction accuracy.