Music Tempo Detection Algorithm Based on Onset Detection

Leyang Wei

CONTENTS

I. ABSTRACT

This report introduces a research project focused on calculating the tempo of music through onset detection. It explains the processing flow, where weighted processing combines the SF and ZCR methods to detect onsets, and calculates the average time interval between onsets to determine the music's BPM. Additionally, the project includes testing with datasets and describes a process for using regular expressions to extract actual BPM values to evaluate the model's accuracy and deviation.

The processing script and dataset used for this report can be obtained from the link:

Here is the link.

II. KEYWORDS

Audio Signal Processing, BPM Detection, Onset Detection, Spectral Flux, Zero Crossing Rate, Dynamic Range Compression.

III. INTRODUCTION

In music analysis, tempo is an important feature, expressed in beats per second (BPM). By analyzing BPM, we can classify music samples based on speed or synchronize music tempo. Humans generally find it intuitive to perceive the beat in music, even when the signals are produced by different instruments or sources. However, it is not easy for a machine to identify these signals. With the advent of digital signal processing technology, researchers have started exploring methods to analyze the rhythm of acoustic music signals. A key challenge in this field is onset detection, which involves identifying the timing of significant events in a signal, such as the beginning of a musical note. Scheirer (1998) proposed an onset detection method based on amplitude variations and spectral differences. This method served as a foundation for numerous subsequent studies on rhythm analysis[1]. The goal of onset detection is to find the starting point of a signal, and it can be used in areas such as speech signal segmentation, sound replacement, and digital effects. Methods to detect it include using spectral flux changes, phase deviations or zero crossing rate changes. This report presents an algorithmic tool for calculating music tempo by detecting changes in spectral flux and identifying onset points in combination with variations in the zero-crossing rate. The algorithm is implemented using MATLAB scripts.

IV. METHODOLOGY

A. Overall Framework

The analysis process involves several steps. First, the average energy interval of the music sample is calculated, and the interval with the highest average energy is selected as the sample for analysis. Next, a band-pass filter and a dynamic compressor are applied to process the sample. The short-time Fourier transform(STFT) is then applied to represent the signal in the time-frequency domain, followed by the calculation of spectral magnitude and zero-crossing rate. Afterward, local maxima are identified, then apply a threshold to filter out unwanted points, resulting in a series of onset points. Ideally, the onset points obtained by both methods would correspond to the beat signals in the music. Their BPM values can then be calculated by averaging the time differences.

B. Core Algorithms

1) RMS calculations: For most music, the most stable part of the temporal distribution is usually found in the chorus part, which typically has a high RMS value. By dividing the music sample into several windows, the RMS value of each window is calculated using equation 1.

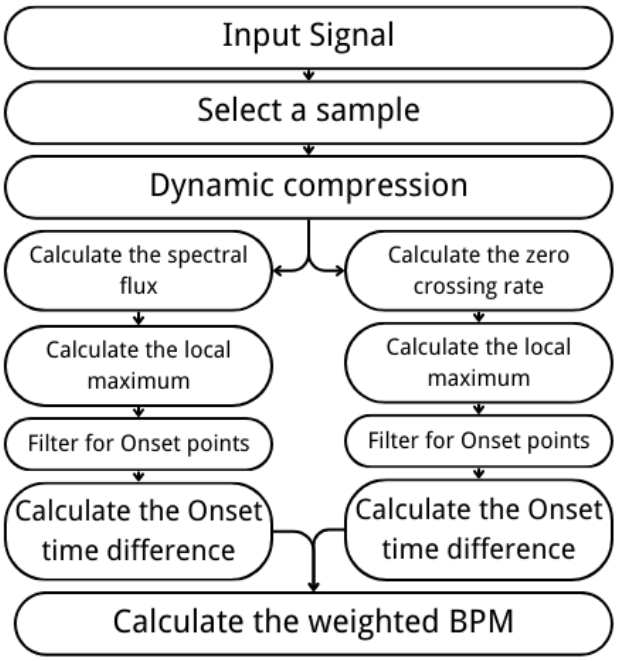

Fig. 1. The main process of calculating BPM

$$\mathrm{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x[i]^2} \tag{1}$$

Where the $x(i)$ represents the i-th window, the root mean square (RMS) of the current window signal is calculated to obtain its RMS value. The interval with the highest RMS value is then selected as the sample for BPM calculation.

2) Spectral Flux (SF) Method: Spectral flux measures the changes in magnitude in each frequency bin and provides the onset function2 , limited to positive changes and sums in the frequency domain.[2]

$$\mathrm{SF(n)} = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} H(|X(n,k)| - |X(n-1,k)|) \tag{2}$$

Where the $H(x) = \frac{x+|x|}{2}$ is the half-wave rectifier function. The specific implementation in MATLAB is the following code,

```
sf = zeros(size(X,2),1);
for n = 2:size(X,2)
    sf(n) = sum(max(X(:,n)-X(:,n-1),0))
        ;
end
```

where `X` represents the absolute value of the input signal after a short-time Fourier transform, meaning that only the amplitude part of the spectrum is retained while the phase part is ignored. After that, `sf` is normalized to obtain a curve with a range of values of 0-1. Next, identify the local maxima of `sf`, using the built-in function `islocalmax`[3] in MATLAB, set the parameter to `MinSeparation`, and specify the minimum separation interval between the maximum values, and generate a logical vector as output. According to the obtained logical vector, the time positions and eigenvalue information of the local maximum values are further extracted. Subsequently, set a threshold to filter the preliminarily extracted local maxima, retaining only the elements that exceed the specified threshold. Finally, two vectors of time positions and corresponding eigenvalue are obtained, that is, the corresponding time and eigenvalue of the onset point. Display it on the graph along with the spectral flux curve to get the figure2. Onset detection using this method offers the advantages of speed and computational simplicity [4]. For rapid detection of large numbers of multiple samples, the detection method based on detecting spectral flux changes will be the preferred method.

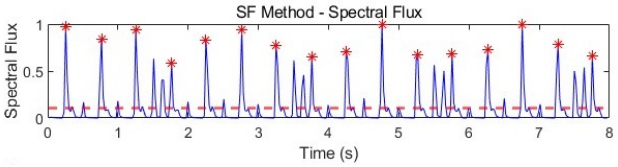

Fig. 2. The onset points obtained by the SF method as well as the spectral flux curves

3) Zero Crossing Rate (ZCR) Method: The ZCR method was introduced to detect onsets in weighted combination with the SF method to improve robustness. The Zero Crossing Rate is calculated by equation3.

$$\mathrm{ZCR}(n) = \frac{1}{N-1}\sum_{i=1}^{N-1} \mathbb{I}(s(i)\cdot s(i-1) < 0) \tag{3}$$

Where the $s(i)$ represents the signal's amplitude at the i-th sample within the n-th frame, while N denotes the total number of samples in the frame being analyzed. The indicator function $\mathbb{I}$ evaluates whether the product of two consecutive samples, $s(i)\cdot s(i-1)$, is less than zero, which occurs when a zero crossing (a sign change from positive to negative or vice versa) happens. The summation aggregates the total number of such crossings within the frame, and dividing by $N-1$ normalizes this count, providing an average zero crossing rate per sample for the n-th frame.

The specific implementation in MATLAB is the following code,

```
Frames = floor((length(X)-windowSize)/
    hopSize)+1;
zcr = zeros(1,Frames);
t = (0:Frames-1)*(hopSize/fs);
for n = 1:Frames
    startIdx = (n-1)*hopSize+1;
    endIdx = startIdx+windowSize-1;
    Frame = X(startIdx:endIdx);
    zcr(n) = sum(abs(diff(sign(Frame)))
        )/(2 * length(Frame));
end
```

Similarly, X represents the absolute value of the input signal obtained through the short-time Fourier transform, where only the amplitude part of the spectrum is retained and the phase part is discarded. The signal is then divided into multiple segments based on the specified window length and step size for windowed calculations. Use the `sign` function to determine the sign of each sampling

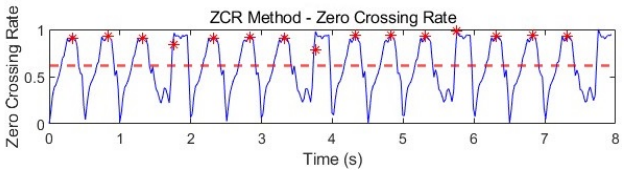

Fig. 3. The onset points obtained by the ZCR method as well as the zero crossing rate curves

point, mapping it to three discrete values: 1, -1 and 0. Then calculate the absolute value of the difference between adjacent sampling points, which means that the difference is zero for the same sign and 2 when the sign changes. Finally, count the interpolations between all change points, divide by the total number of sampling points, and then divide by 2 to calculate the zero-crossing rate for the current window. Store the ZCR values of all windows in a vector to generate the ZCR characteristic curve.

After that, as before, `zcr` is normalized to obtain a curve with a range of values of 0-1. The local maximum values of the `zcr` curve are identified using MATLAB's built-in `islocalmax` function. Based on the resulting logical vector, the time positions and feature values of the local maxima are extracted. Subsequently, a threshold is then applied to filter these preliminary maxima to obtain two vectors: one representing the time positions and the other representing the corresponding feature values of the onset points. These onset points, along with the spectral flux curve, are then displayed on the graph to produce the final figure4.

4) Dynamic Range Compression: In order to exclude interference from signal dynamics, a dynamic compressor is used to dynamically compress the dynamic range of the sample. Using equation4 to compress the dynamic range of the signal.

$$x_{\text{out}}(n) = \begin{cases} x(n), & \text{if } |x(n)| \leq T \\ \text{sign}(x(n)) \cdot \left(T + \frac{|x(n)|-T}{R}\right), & \text{if } |x(n)| > T \end{cases} \tag{4}$$

Where T is the compression threshold. When the absolute value of the signal $|x(n)|$ is less than or equal to T, the output signal $x_{\text{out}}(n)$ remains unchanged. However, if $|x(n)| > T$, the signal will be compressed, where R represents the compression ratio. The function $\text{sign}(x(n))$ preserves the original polarity of the signal, ensuring that the positive and negative values of the output correspond to those of the input.

5) Band-pass filter: Before performing onset detection, apply a band-pass filter to the signal to reduce interference from low-frequency and high-frequency noise. For both the SF method and the ZCR method, different frequency ranges need to be set to suit their respective detection intervals.

The filter uses a Butterworth filter design to provide a flat frequency response curve. In Matlab, a built-in Butterworth filter design function is provided, which returns the transfer function coefficients of the `n`-order Butterworth filter with a normalized cutoff frequency of `Wn`.

```
[b, a] = butter(2, frequencyRange / (fs
    / 2), 'bandpass');
x = filter(b, a, x);
```

Where the `frequencyRange` is a binary vector that defines the lowest and highest cutoff frequencies of the band-pass filter, and divides them by the Nyquist frequency to normalize the frequency range. Afterwards, the function returns the coefficients of the transfer function. The vectors `b` and `a` are passed to the `filter` function to obtain the filtered signal.

6) BPM calculations: Ideally, the onset points obtained by the SF and ZCR methods correspond relatively accurately to the beat signal in the music. For a single calculation method, the standard deviation can be used to filter out abnormal onset points and then calculate the time difference between the remaining points to calculate the relatively accurate BPM value. The specific process is as follows5:

$$\Delta T = t_{i+1} - t_i, \tag{5}$$

First, the time difference vector between the onset points is calculated. The symbol ΔT represents the set of time differences, where Δt_i denotes the time difference between two consecutive onset points, calculated as $t_{i+1} - t_i$. Next, calculate the standard deviation using equation 6 for filtering.

$$\begin{aligned} \mu_{\Delta T} &= \frac{1}{N}\sum_{i=1}^{N} \Delta t_i, \\ \sigma_{\Delta T} &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\Delta t_i - \mu_{\Delta T})^2} \end{aligned} \tag{6}$$

The symbol $\mu_{\Delta T}$ represents the mean of the time difference set ΔT, calculated as the total sum of the time differences divided by the number of time differences N. The symbol $\sigma_{\Delta T}$ denotes the standard deviation of the time difference set, which measures the dispersion of time differences, defined as the square root of the mean of the squared deviations from the mean7.

$$\Delta T_{\text{filtered}} = \{\Delta t_i \mid |\Delta t_i - \mu_{\Delta T}| \leq k \cdot \sigma_{\Delta T}\} \tag{7}$$

The outliers are then filtered using the standard deviation. The symbol $\Delta T_{\text{filtered}}$ represents the filtered set of time differences, where each element Δt_i satisfies the condition $|\Delta t_i - \mu_{\Delta T}| \leq k \cdot \sigma_{\Delta T}$. Here, $\mu_{\Delta T}$ denotes the mean of the time differences, $\sigma_{\Delta T}$ represents the standard deviation of the time differences, and k is a factor controlling the filtering range.Finally, use equation 8 to calculate the BPM value.

$$\begin{aligned} \bar{\Delta} t &= \frac{1}{M}\sum_{i=1}^{M} \Delta t_i, \\ \text{BPM} &= \frac{60}{\bar{\Delta} t} \end{aligned} \tag{8}$$

The symbol $\bar{\Delta} t$ represents the average value of the filtered set of time differences, calculated as the total sum of the time differences divided by their count M. The symbol BPM denotes the beats per minute, calculated by dividing 60 by the average time difference $\bar{\Delta} t$.

In the algorithms discussed in this article, two methods were used to calculate the weighted BPM, and the above process calculated the BPM values obtained by individual methods. Next, calculate the weights of each method to compute the weighted results. The weights are calculated using the standard deviations of the filtered onset sets obtained from the two detection methods. 9

$$\mu_{\Delta T_{\text{filtered}}} = \frac{1}{|\Delta T_{\text{filtered}}|} \sum_{i=1}^{|\Delta T_{\text{filtered}}|} \Delta t_i,$$
$$\sigma_{\Delta T_{\text{filtered}}} = \sqrt{\frac{1}{|\Delta T_{\text{filtered}}|} \sum_{i=1}^{|\Delta T_{\text{filtered}}|} (\Delta t_i - \mu_{\Delta T_{\text{filtered}}})^2}. \tag{9}$$

The symbol $\sigma_{\Delta T_{\text{filtered}}}$ represents the standard deviation, quantifying the spread of the time differences Δt_i within the filtered set $\Delta T_{\text{filtered}}$ around their mean $\mu_{\Delta T_{\text{filtered}}}$. The calculation involves two steps. First, the mean $\mu_{\Delta T_{\text{filtered}}}$ is obtained as the average of all time differences in the filtered set $\Delta T_{\text{filtered}}$. Second, $\sigma_{\Delta T_{\text{filtered}}}$ is determined as the square root of the average squared deviations of Δt_i from the mean $\mu_{\Delta T_{\text{filtered}}}$, providing a measure of how far the time differences in the filtered set deviate from the average. Finally, the respective weights are determined 10.

$$\begin{aligned}
U_{SF} &= \frac{1}{\sigma_{SF}}, \quad U_{ZCR} = \frac{1}{\sigma_{ZCR}}, \\
U_{\text{total}} &= U_{SF} + U_{ZCR}, \\
w_{SF} &= \begin{cases} 1, & \sigma_{SF} = 0, \\ \frac{U_{SF}}{U_{\text{total}}}, & \sigma_{SF} > 0 \text{ and } \sigma_{ZCR} > 0, \\ 0.5, & \sigma_{SF} = 0 \text{ and } \sigma_{ZCR} = 0, \end{cases} \\
w_{ZCR} &= \begin{cases} 1, & \sigma_{ZCR} = 0, \\ \frac{U_{ZCR}}{U_{\text{total}}}, & \sigma_{SF} > 0 \text{ and } \sigma_{ZCR} > 0, \\ 0.5, & \sigma_{SF} = 0 \text{ and } \sigma_{ZCR} = 0, \end{cases} \\
\text{BPM} &= w_{SF} \cdot \text{BPM}_{SF} + w_{ZCR} \cdot \text{BPM}_{ZCR}.
\end{aligned} \tag{10}$$

Where U represents the reciprocal of the standard deviation, representing consistency: the larger the standard deviation, the greater the error, and thus the smaller the consistency. The weights are calculated by dividing each U value by their total sum. Finally, the combined BPM value is obtained as the weighted sum of the BPM values from the SF and ZCR methods. When the standard deviation of one side is zero, it indicates that the data of the corresponding method has no deviation. In this case, a weight of 1 can be fully assigned to this method. When both standard deviations are zero, the weights of both methods are set to be equal.

However, in some extreme cases, due to the complexity of the signal, a portion of the beat signal is submerged in the noise signal and cannot be detected, resulting in the calculated BPM value being too small compared to the actual value (usually in a multiplier relationship). Therefore, based on robustness considerations, incorporating judgment logic and multiplying excessively small values by appropriate factors (e.g., 2 or 4) ensures that the calculated BPM values fall within the typical range (e.g., 60–180 BPM).

V. Verification and Discussion

A. Data set

The EDM-TR9 dataset[5] was used for performance testing, The dataset is an unmodified original dataset. EDM-TR9 is an open audio dataset consisting of a series of electronic drum recordings. This dataset mainly focuses on the unique sound and rhythm patterns of the Roland TR-909 drum machine in dance, chamber music, and electronic music sub genres. This dataset contains 3780 audio loops recorded in uncompressed stereo WAV format, created using custom drum samples and MIDI programmed rhythms at various rhythm rates.

B. Verification method

For verification, the regular expression is used to extract the real BPM value from the file name of each sample in the data set. For example, if the file name format of the real BPM contained in each file name in the dataset is `"120bpm_tr9_drm_id_001_0001.wav"`, the numbers in the file name are extracted by regular expressions as the real BPM values.

```
pattern = '(\d+)bpm';
tokens = regexp(fileName, pattern, '
    tokens');
realBPM = str2double(tokens{1}{1});
```

where `pattern` defines the regular expression pattern and matches the number part of the string. `tokens` indicates an array of matching results, using the regex matching function `regexp` in MATLAB to specify the file name and regex mode, and `'tokens'` specifies that only the contents of the capture group in the regex are returned. Finally, the `str2double` function in MATLAB is used to convert the string to a numeric value, and the `tokens{1}{1}` represent the string content in the capture group, where the string content is the BPM value.

C. Octave error

Octave error refers to a situation where the estimated BPM value is a multiple or fraction of the actual value. This phenomenon is closely related to the subjectivity of rhythm perception, as individuals may interpret the tempo of the same piece of music differently[6]. Therefore, three verification methods are set: Accuracy1, Accuracy2, and Accuracy3.

Accuracy1:Compare only the true BPM values annotated in the dataset.

Accuracy2:Compare values of the true BPM annotated in the dataset with their double and half values.

Accuracy3:Compare twice, 1.5 times, half and 1/1.5 times the actual BPM values.

When verifying using Accuracy2 and Accuracy3 methods, the calculated BPM value is compared with the nearest value in the actual BPM array, if the difference is within a predefined threshold, the detection is deemed correct; otherwise, a detection error is recorded. Finally, a dichotomous confusion matrix is generated to represent the number of correct and incorrect detections. In addition, the deviation rate for each calculated BPM value is determined

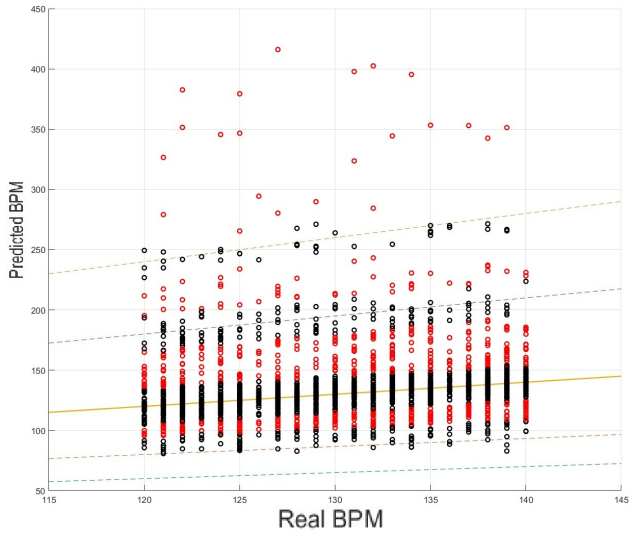

Fig. 4. Dataset test results

by comparing it with the closest real BPM value. The average deviation rate is then computed across all samples to provide an overall performance metric.

Among the 3780 samples, according to the Accuracy3 verification method, it was found that the number of samples with numerical differences less than 15 accounted for 84% of the samples, and some samples with clear and stable rhythms had a deviation rate close to 1% or lower. According to Accuracy2 and Accuracy1, the accuracy was decreased to 77%. By plotting the results of each sample on a figure with the true values as the X-axis and the detected values as the Y-axis, the following figure is obtained4:

Where the five straight lines in the figure correspond to the true BPM values and their octave value, the solid lines represent the values annotated on the dataset, and the others represent 2 times and 1.5 times values (from top to bottom, they are 2 times, 1.5 times, original value, 1/1.5 times and 1/2 times, respectively). Samples exceeding the tolerance deviation value are marked in red. The following tableI is obtained by counting the samples with correct and incorrect detections.

TABLE I
STATISTICAL TABLE OF INCORRECT AND CORRECT SAMPLES

	Accuracy1	Accuracy2	Accuracy3
Number of samples	2911	2911	3188
Percentage	77.01%	77.01%	84.34%

Combining the table and chart, it can be seen that the number of samples marked as correctly detected is much larger than the number of samples marked as incorrect, and most of them fall on the straight line of true BMP values, while a small portion falls on the straight line of octave values.

D. Limitations

Although the data looks promising, the decrease in accuracy when the verification method was switched from Accuracy3 to Accuracy2 revealed a 1.5 octave error. This indicates that some beat signals, particularly those corresponding to 6 or 3 beats, were missed. Such an issue cannot be ignored.

Furthermore, errors in some incorrectly detected samples are primarily attributed to missed onset detection. In these cases, some beat signals are too weak to be identified. For samples with unstable rhythms and more complex arrangements, the accuracy rate is notably lower. This highlights the inherent limitations of this method in detecting music rhythm, as its accuracy heavily relies on the appropriate configuration of thresholds and local maximum detection windows.

VI. Potential areas for improvement

Refer to a variety of methods to make comprehensive judgments. The method discussed in this paper focuses on the change in the frequency of the sample to analyze the tempo velocity and ignores the change in the envelope of the sample. Therefore, combined with the audio envelope change obtained by rectification and low-pass filtering, it may be feasible to comprehensively evaluate the beat speed of music through an autocorrelation algorithm [7].

In the context of the rapid development of artificial intelligence and big data, the introduction of machine learning methods may be a feasible solution. It may be a feasible direction to pre-judge the velocity range of the sample according to different music eigenvalues, and set an appropriate detection window or threshold.

References

[1] A. Klapuri, A. Eronen, and J. Astola, “Analysis of the meter of acoustic musical signals,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 342–355, 2006.

[2] S. Dixon, “Onset detection revisited,” in *Proceedings of the 9th International Conference on Digital Audio Effects (DAFx’06)*, Montreal, Canada, September 18–20 2006, p. 2.

[3] I. © 1994-2024 The MathWorks, “islocalmax function documentation,” 2025, accessed: 2025-01-01. [Online]. Available: https://nl.mathworks.com/help/matlab/ref/islocalmax.html

[4] S. Dixon, “Onset detection revisited,” in *Proceedings of the 9th International Conference on Digital Audio Effects (DAFx’06)*, Montreal, Canada, September 18–20 2006, p. 5.

[5] Patchbanks, “Waivops edm-tr9: Open audio resources for machine learning in music,” Patchbanks, Dec 2023. [Online]. Available: https://doi.org/10.5281/zenodo.10278066

[6] A. Iliadis, D. Tsipianitis, A. Paraskevas, K. Tsaprailis-Chablas, and P. S. Trakas, “On beat tracking and tempo estimation of musical audio signals via deep learning,” in *2023 14th International Conference on Information, Intelligence, Systems Applications (IISA)*, 2023, p. 3.

[7] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, “An experimental comparison of audio tempo induction algorithms,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1832–1844, 2006.