TemPoRe: Tempo Tracking in Real-time with Polynomial Regression

First Author
Affiliation1
author1@myorg.org

Second Author
Affiliation2
author2@myorg.org

Third Author
Affiliation3
author3@myorg.org

ABSTRACT

TemPoRe *is an algorithm implemented in Python that combines traditional methods with a machine learning technique to track tempo in real-time. To this end, the autocorrelation function is calculated on the incoming signal to detect tempo and polynomial regression is utilized to predict the next tempo based on the prior tempo to obtain a more stable tempo. This paper will demonstrate how supervised learning can improve the accuracy of real-time tempo tracking and additionally suggest ways to further enhance the performance of this algorithm.*

1. INTRODUCTION

Tempo is one of the most important elements of music. It is therefore not surprising that a lot of research has been done on tempo tracking to date. DAW programs, such as Logic and Ableton Live easily find tempo in loaded files, however, a means of accurately detecting tempo in real-time scenarios isn't often found in such programs as a built-in function. To track tempo in real-time, it is necessary to determine whether the current note onset is a tempo determining beat; this is usually decided through the strength of the onset. But it is difficult to know how strong it is relative to the other onsets without looking at the entire piece, which might not be available in advance in a real-time scenario.

TemPoRe aims to address these problems of estimating tempo in real-time, making it more robust by combining an algorithm for detecting tempo with a machine learning technique for predicting tempo. The reason for using the prediction method is that humans tend to anticipate the next beat based on the prior beat when perceiving a beat [1]. Therefore, it is reasonable to apply this method to the tempo tracking algorithm. This paper will first describe previous studies related to tempo tracking and subsequently the two methods used in *TemPoRe*. Finally, I will discuss how to decide a global tempo from the two tempi obtained by the two methods.

2. BACKGROUND

To detect tempo, it is essential to track the beat first. This is because tempo can only be determined once we know which note onset is the beat. Tempo tracking and beat tracking are classic research topics in Music Information Retrieval (MIR) tasks, which require a comprehensive understanding of various fields such as music, computer-based technology, and psychology. In recent decades there has been a substantial amount of wide-ranging research conducted on these topics.

E. D. Scheirer's research [2] can be considered as one of the pioneering studies in beat tracking [3]. In his work, he divides the signal into its frequency bands using a comb filter and tracks tempo by observing where the energy spike increases the most when looking at the amplitude. Although he indicated in his research that he could not specify the right features of the tempo tracker, he focused on the similarity between rhythm tracking and pitch tracking and adopted an efficient approach of using filter banks and autocorrelation for rhythm tracking.

Another approach to tempo tracking can be found in B. Pardo's research [4]. His study uses a single oscillator to track note onsets that occur within a given window of time and applies weights averaged previous tempo periods to the current estimated tempo periods to arrive at the global tempo. However, achieving high accuracy using this method was difficult because the algorithm only considers the weight of the tempo period, and there is no way to distinguish which of the note onsets is the beat since note onsets are not separately classified.

Subsequently, D.P.W. Ellis's model [5] transforms a preloaded sound file into an onset-strength envelope and calculates the autocorrelation function of the signal to track beats of the rhythm. He additionally utilizes dynamic programming to obtain the globally-optimal beat sequence. His model has been verified to accurately detect tempo, and the method is utilized as the algorithm for the `librosa.beat.track` class of the Librosa library [6] in Python, which is currently one of the most commonly used beat tracking methods.

In the current research on beat tracking, Steinmetz and Reiss's model [7] can be considered a state-of-the-art study. They predict beats and downbeats directly from waveforms without pre-processing, via methods such as spectral engineering, by an end-to-end approach. Temporal Convolutional Networks (TCN) are utilized for this model to achieve a large receptive field. This research could be a good example of combining MIR and deep learning, which is currently an active area of research.

3. TWO METHODS OF DETECTING TEMPO

Real-time tempo tracking, an extension of beat-tracking, poses various difficulties. First, one must identify which onset serves as the beat, the periodic time interval between onsets, even if multiple notes appear simultaneously. To this end, it is usually done by finding strong peaks among onsets and choosing an appropriate beat based on the music's overall average tempo period. While this process may not be difficult for an already loaded file, since we can pick strong peaks for each section by looking throughout the music ahead of time, it is challenging to determine the strength of the current peaks relative to the upcoming parts in real-time. Therefore, finding the beat is not an easy task when processing an input signal in real-time.

TemPoRe employs two distinct methods, both of which are implemented in Python, which are combined to track tempo more stably and accurately in real-time. The first method, inspired by Ellis's work, above, utilizes the autocorrelation function to estimate tempo and the second method uses polynomial regression to predict the next tempo based on the prior tracked tempo. The autocorrelation function is obtained using the `librosa.feature.tempo` class of the Librosa library [8], and polynomial regression is performed using the `sklearn.linear_model` class of the Sklearn library [9]. While the fundamental idea of this paper is to track tempo in real-time, an audio file was used to evaluate the algorithm instead of a real-time microphone input, in order to avoid the potential variability caused by microphone type, performance, or other environmental factors. Nonetheless, *TemPoRe* itself operates in real-time as if the sound file were a live sound source.

3.1 The Autocorrelation Function

Among the basic techniques for beat tracking and tempo estimation described above, such as those utilizing oscillators and comb filters, the autocorrelation function is the base algorithm selected for use in *TemPoRe*. This technique identifies strong peaks by getting the signal's own similarity. This involves transforming the signal into the onset-strength envelope, and then sequentially multiplying itself at delayed time points, thus emphasizing the strong peaks and identifying them as the basis for the tempo estimation.

The `feature.tempo` class is used to estimate tempo from an incoming signal. The algorithm of this class can be broken down into three main steps. First, the bpm is calculated for each bin using the onset-strength function within the Librosa library, which is followed by weighting using the autocorrelation function. Finally, any tempo exceeding the maximum tempo is discarded, and the weight is adjusted by multiplying with a prior value before calculating the maximum tempo [8].

To track tempo in real-time, *TemPoRe* needs to evaluate the local tempo of smaller segments of sound, not the global tempo of an entire sound file. Although the `beat.track` class is also available for tempo tracking in the Librosa library, it was not used in this paper because the `feature.tempo` class performed better in tracking the tempo over short segments of the sound when both classes were tested and compared. The `beat.track` class also translates signals into the onset-strength envelope and calculates the autocorrelation function to track the tempo, but differs from the `beat.tempo` class in selecting the beat based on peak tracking. Additionally, the `feature.tempo` class calculates the weight by multiplying the prior value, whereas the `beat.track` class assumes that tempo is relatively constant throughout the music and picks the strong peak closest to the estimated tempo as the reference for the strong beat, which is then used to calculate tempo [6].

In an attempt to utilize the `feature.tempo` class for real-time tempo tracking, I set the window size to four seconds and receive the input signal four times every second, as shown in Figure 1. From the four tempo values obtained from each of the four windows, the maximum and minimum values are discarded and the average of the remaining two values is calculated. The result of this process is then determined as the 'estimated tempo'. This proposed approach aims to mitigate the problem of sudden and large tempo variations, which can lead to inaccurate tempo tracking. By utilizing this process, it was possible to make the tempo change smoothly and steadily.

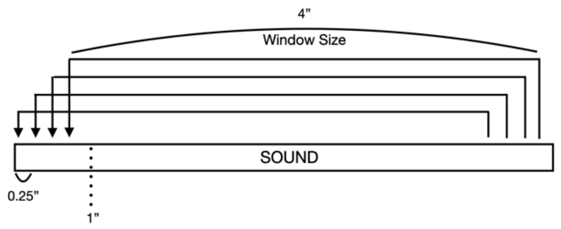


Figure 1. Four windows receiving the input signal

3.2 Polynomial Regression

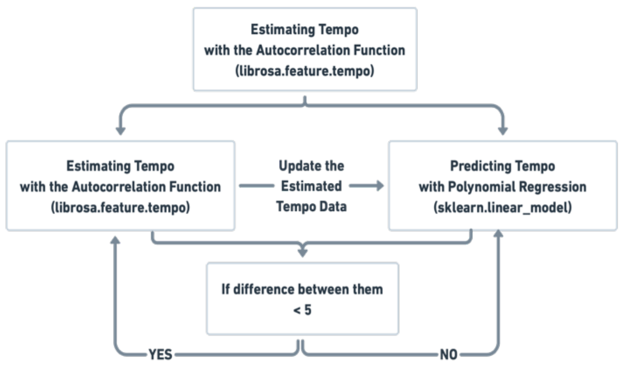
Using only the estimated tempo obtained by the autucorrelation function can lead to various problems when detecting tempo. If only the estimated tempo is used, there is the possibility of incorrectly detecting the beat in certain situations, such as trills or long pauses (i.e, rests) in the music. Therefore, I addressed these possible errors by using a combination of the estimated tempo and the 'predicted tempo' obtained by anticipating the tempo based on the prior estimated tempi.

In this study, polynomial regression is utilized to predict the tempo. Polynomial regression is a type of linear regression that learns the correlation between independent and dependent variables and predicts the next value based on it. The difference between polynomial regression and linear regression is that while linear regression can only obtain predicted values in the form of a straight line, polynomial regression can obtain predicted values in the form of a curve with multiple degrees using a polynomial. Predicting tempo using a linear approach is difficult since tempo does not generally increase or decrease uniformly. Therefore, polynomial regression is used to increase accuracy.

Tempo is detected every two seconds using the `feature.tempo` class and this data is used as input for polynomial regression to predict the next tempo. That is, the data for polynomial regression is updated every two seconds, and the machine also learns and performs every two seconds. The `sklearn.linear_model` class is used for this process. After testing various values, a degree of 4 was found to be the most suitable for *TemPoRe*.

4. CHOOSING THE GLOBAL TEMPO

It is important not only to find the tempo through two methods – the autocorrelation function and polynomial regression – but also very important how to determine which of these tempi serves as the global tempo. This is because the accuracy of tempo tracking can vary depending on which tempo is selected. To this end, the determination of the global tempo in *TemPoRe* involves a two-step process that is executed alternately every second shown in Figure 2.



Estimating Tempo with the Autocorrelation Function (librosa.feature.tempo)
Estimating Tempo with the Autocorrelation Function (librosa.feature.tempo)
Update the Estimated Tempo Data
Predicting Tempo with Polynomial Regression (sklearn.linear_model)
If difference between them < 5
YES
NO

Figure 2. Flowchart of the *TemPoRe* algorithm.

In the first step, in the upper part of the figure, the estimated tempo obtained through the autocorrelation function is always selected, while simultaneously calculating the difference between the estimated tempo and the predicted tempo obtained through polynomial regression. In the second step, in the lower part of the figure, the decision to use either the estimated tempo or the predicted tempo is made based on the difference between their values. If the difference is less than 5, the last estimated tempo is used as in the first step, and conversely, if it is greater than 5, the predicted tempo is used instead. This approach aims to enhance the robustness of the global tempo by alternately choosing between the two tempi depending on the difference, and thereby avoiding radical changes in the tempo.

5. RESULTS

To verify the accuracy of *TemPoRe*'s performance, a comparison was made between the data obtained from professionally trained musicians tapping along with a beat, and the data obtained from *TemPoRe*. Bartok's <Pentatonic Melody> No. 61 from Mikrokosmos Vol. 2 [10] was used as the test music. Although this piano piece has a relatively simple structure, it contains diverse rhythm patterns, including syncopation, making it suitable for testing the accuracy of *TemPoRe* in various situations.

5.1 Tapping Data

Cognitive data from actual people seemed to be the most suitable data to serve as test data for comparison with *TemPoRe*. The group of ten people listened <Pentatonic Melody> No. 61 and were asked to tap where they felt the beat, and the average value of this data was calculated. The result of comparing this tapping data and the tempo obtained through *TemPoRe* is shown in Figure 3. The solid line represents the tempo of the tapping data obtained through the group of musicians, and the dots indicate the global tempo obtained through *TemPoRe*.

The most significant difference was found around 45 seconds into the piece, where *TemPoRe* detected the tempo as 88 bpm, whereas the tapping data indicated 96 bpm at this point in the music. This section corresponds to bar 31 in the score of <Pentatonic Melody>, where the performer played an expressive *accelerando*. Although *TemPoRe* also detected that the tempo of that part was increasing, it could not estimate the sudden and fast changes as accurately as humans did.

On the other hand, *TemPoRe*'s accuracy was highest in the first 10 seconds and between 25-35 seconds, where it tracked the tempo almost identically to the test data. Furthermore, it was concluded that *TempPoRe* detected *ritardando* quite well in the part where the tempo suddenly became faster and then slowed down abruptly at the end of the piece. When comparing the two tempi, it was found that there was a slight difference in certain parts, but the direction of the tempo such as speeding up or slowing down was well tracked in general.

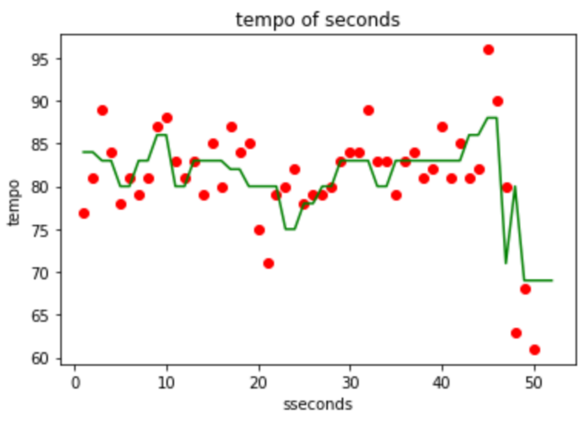


Figure 3. The comparison of the tapping data of musicians and the data of TemPoRe

6. CONCLUSIONS

This research suggests a method for combining machine learning techniques with traditional computational methods from MIR research in order to improve the accuracy of the results. *TemPoRe* utilizes the autocorrelation function and polynomial regression to obtain the estimated tempo and the predicted tempo, respectively. By using both tempo values alternately, the goal is to track the tempo more stably in real-time. As a result, it can be shown that such a hybrid approaches able to track tempo relatively well in pieces like Bartok's <Pentatonic Melody>.

However, the accuracy decreases when notes fluctuate with *accelerando* or when there are many embellishments such as trills, tremolo, etc. This issue can be solved by applying the estimating tempo algorithm only to specific ranges of the spectrum where note onsets occur, instead of tracking the tempo for the entire incoming signal. This shows the necessity of developing methods for spectrogram-based analysis and effectively applying them to *TemPoRe* in the future.

The next step of this study is to implement *TemPoRe*, which is currently only available in Python, into MAX/MSP. Since many musicians use MAX/MSP for creating electronic music, this implementation will make *TemPoRe* more widely accessible and valuable. Moreover, it is anticipated that this method will also prove to be valuable for Human-Algorithm Ensemble, Automatic Music Transcription (AMT), and other related functions.

7. REFERENCES

[1] J. A. Grahn, J. B. Rowe, "Finding and feeling the musical beat: striatal dissociations between detection and prediction of regularity," *Cerebral cortex*, vol. 23, no.4, pp.913-921, 2013.

[2] E. D. Scheirer, "Pulse tracking with a pitch tracker," *Proceedings of 1997 Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, 1997, pp.1-4.

[3] D. Deutsch, *The Psychology of Music*. Elsevier Science, 2012.

[4] B. Pardo, "Tempo Tracking with a Single Oscillator," *Proceedings of International Conference on Music Information Retrieval (ISMIR)*, Barcelona, 2004, pp.154-157.

[5] D.P.W. Ellis, "Beat Tracking with Dynamic Programming," *Journal of New Music Research*, vol. 36, no.1, pp. 51-60, 2006.

[6] "librosa.beat.track," https://librosa.org/doc/main/generated/librosa.beat.beat_track.html, accessed April 11, 2023

[7] C. J. Steinmetz, and J. D. Reiss, "WaveBeat: End-to-end beat and downbeat tracking in the time domain," *Proceedings of the 151st Audio Engineering Society Convention, Las Vegas*, 2021. https://www.aes.org/publications/proceedings/?num=151, accessed April 17, 2023.

[8] "librosa.feature.tempo," https://librosa.org/doc/main/generated/librosa.feature.tempo.html, accessed April 10, 2023.

[9] "sklearn.linear_model," https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html, accessed January 9, 2023.

[10] B. Bartok, *<Pentatonic Melody> No. 61 from Mikrokosmos Vol. 2.* https://www.youtube.com/watch?v=ML6LXzBUPdk, accessed March 15, 2023.