Automated Musical Tempo Estimation with Neural-Network-Based Onset Detection

Assessing Onset Detection Quality in Tempo Estimation

Nathan Stephenson

Thomas Jefferson High School for Science and Technology
Computer Systems Lab 2020–2021

Abstract

When synchronizing media to music and comparing song features, the tempo or speed of a song is a highly important piece of information to have. Using a tempo estimation method detailed by van de Wetering, musical tempo can be determined with high accuracy through automated means. This method interprets the onsets of a song, which are the starts of sounds or musical notes. In this paper, various tempo detection methods as well as various onset detection methods are assessed to determine which are the best fit for precision and accuracy. Complex Domain and LL are found to be the most well-rounded onset detection methods due to their high accuracy and also as they can both be run in real-time. Further data should be collected to pinpoint the specific strengths and weaknesses of each onset detection method.

1 Introduction

Tempo detection is an important technology for the synchronization of anything to music. Knowing the rate at which beats occur per minute makes it significantly more convenient to time events in music-based games and videos, and an automated method to find the beats per minute (BPM) of an audio track is especially important when precision beyond the human ear is necessary and when BPM needs to be calculated for more than a few songs, as manually determining the BPM of a track using a metronome and trial and error is tedious.

For this project, I am focusing only on music with a constant tempo. Constant tempo was not common in music until the popularity of digital audio workstations and other forms of digital music production increased, meaning that artists such as the Beatles and many artists before the 1980s made music with variable tempo values. In those cases, a single value would not be enough to match with the statistical purposes. Nonetheless, I will also provide examples of music with variable BPM values.

2 Background

Existing development of state of the art tempo detection is primarily led by Sebastian Böck, who has worked on improving the detection of onsets and beats using neural networks under the Austrian Research Institute for Artificial Intelligence. Onsets refer to the start of a sound in a musical waveform, and through these onsets the actual beats in the audio are determined, and using the beats we can find the tempo (in beats per minute). In his diploma thesis [2], Böck explores several methods of onset detection, as well as many methods of beat tracking such as comb filter and histogram based tracking. He also proposes a new method of onset detection based on neural networks, referred to as bidirectional Long Short-Term Memory (BLSTM).

In 2012, Böck *et al.* proposed an onset detection method that uses unidirectional recurrent neural networks (RNNs) [3] instead of bidirectional ones as seen with the BLSTM approach. This method maintains high accuracy but also due to its unidirectional nature is capable of detecting onsets in real time, an ability that the BLSTM approach lacks. The next year, Jan Schlüter and Böck presented an onset detection method that utilizes convolutional neural networks (CNNs), which was found to slightly surpass the BLSTM approach in its F-score at the cost of performance [4]. Both of these onset detection have been implemented into the `madmom` Python library for audio statistics.

In 2015, Böck et. al proposed a comb filter-based tempo detection method which provided 94.6% total average accuracy, a 1.7% increase over the best of six state-of-the-art tempo detection methods [5], and this was incorporated into `madmom` as well.

However, a fairly unknown report was written by van de Wetering [1] on tempo detection, intended for the purpose of synchronizing music to rhythm games. Rhythm games require the player to play accurately to the music, and as such the game needs to be able to determine if the player's actions match with the music, making it important that the tempo used in the game is accurate. I found van de Wetering's method in a program he wrote called ArrowVortex, which allows one to create rhythm game "charts," meaning synchronized instructions and patterns. Van de Wetering's solution in ArrowVortex provides promising results in comparison to Böck's method in `madmom`, presenting a 99.982% accuracy on a small sample of data, beating `madmom`'s 99.257% accuracy by 0.725%. As seen in Table 1, *Perm* is an upbeat funk song by Bruno Mars with clear drum breaks, *Move Your Feet* is a dance-pop song by Junior Senior with an oddly specific decimal tempo, *STAR LINER* by A-One is a Japanese eurobeat with loud percussion and synth lines, and *Toulouse Street* is an acoustic pop rock song with layered guitars and a variable tempo.

Accuracy should be as close to 100% as possible for as many songs as possible in order to properly synchronize anything to music, or else the rhythm will slowly diverge. Van de Wetering's solution seems to show accurate results under the assumption that the tempo is constant in the music (which is true of most modern, professionally produced music) and that there are relatively sharp changes in the waveform (making slow orchestral works a bad fit, for example).

According to Böck [5], the F-measure (measure of accuracy) of BLSTM is significantly

Method	Perm	Move Your Feet	STAR LINER	Toulouse Street
Van de Wetering (ArrowVortex)	124	118.868	162	179.402
Böck et. al (`madmom`)	61.86 (123.72)	59.41 (118.82)	162	89.55 (179.1)
Actual	124	118.879	162	90 (180)[1]

[1] This tempo is variable and an average integer tempo is suggested.

[2] BPM values with parentheses are the originals multiplied by 2 to match other values, as multiplying or dividing BPM values by powers of 2 do not change the synchronization of the beats.

Table 1: Comparison of various tempo detection methods on four songs.

higher than Spectral Flux, which is used in van de Wetering's report, and newer onset detection methods that utilize neural networks could further improve the consistency of van de Wetering's method, especially when beats are less pronounced in the audio or are hard to detect.

3 Methods

In order to assess the performance of various tempo and onset detection methods on a reliable source of music with fixed tempo, a dataset consisting of contemporary music is ideal as a large portion of professionally-recorded music is digitally produced and is likely to have a constant tempo. In this study, a selection of 30-second samples of music were taken from Spotify for reference. The artists were selected mostly through Billboard's Top Artists of the 2010s list, though some extra artists were also added into the selection. The nature of the dataset suggests that the majority of the music being tested falls under pop or rap music, but genres such as country and EDM are also incorporated. Ground truth is not assessed manually for this dataset; the mode is used as ground truth as a consensus is most likely to be the correct value for a given tempo. If there is no mode, then the sample is not used in the calculations as it likely is part of a song with a variable BPM.

Using van de Wetering's tempo detection process as a base, I will compare various onset detection methods to see which works the best with the implementation. Nine algorithms will be taken from the `aubio` Python library as listed in its documentation. Three more algorithms are taken the `madmom` Python library, which I will refer to as BLSTM, CNN, and LL (Online Real-Time RNN). These algorithms are referenced in the previous section.

Some changes were made from van de Wetering's code implementation in order to match the high precision and accuracy of ArrowVortex's tempo detection; the main one being that onset strengths were originally calculated and evaluated, but taking onset strengths into account actually harm the results and the replication of high accuracy. As such, each onset is given a constant strength value during calculations.

Other tempo detection methods will be included as well; both `aubio` and `madmom` provide their own tempo detection methods which can be compared to van de Wetering's solution. One more tempo detection method will be included, which is Spotify's own tempo detection algorithm, accessible through their API and included with song data; thus, the songs that are in the Spotify dataset will have a corresponding BPM already associated with them by Spotify.

4 Results

Method[1]	**Perm**	**Move Your Feet**	**STAR LINER**	**Toulouse Street**
BLSTM	124	118.879	162	179.402
Complex	124	118.879	162	179.378
CNN	124	118.879	162	179.817
Energy	124	118.879	162	179.841
HFC	124	118.879	162	179.378
LL (RNN)	124	118.879	162	179.817
MKL	124	118.879	162	179.366
KL	124	118.879	162	179.817
PB	124	118.879	189.596	179.390
SD	124	118.879	162	179.817
SF	124	118.879	162	179.329
Actual	124	118.879[2]	162	90 (180)[3]

[1] Italicized methods are neural-network-based.
[2] This tempo is variable and an average integer tempo is suggested.
[3] BPM values with parentheses are the originals multiplied by 2 to match other values, as multiplying or dividing BPM values by powers of 2 do not change the synchronization of the beats.

Table 2: Comparison of various onset detection methods on four songs using van de Wetering's implementation as a basis.

Table 2 shows the results of using different onset detection methods in tandem with van de Wetering's tempo detection process. Most of the onset detection methods had no trouble determining the tempo of fixed BPM songs with a clear beat, with the exception of the PB (Phase-Based) method which incorrectly determined STAR LINER's BPM; it is possible that the high amount of percussion in STAR LINER made it difficult to discern the primary rhythm correctly. Nonetheless, these values demonstrate the resilience of van de Wetering's implementation despite variance in the quality of onset detections.

Toulouse Street hovers around 180 BPM, and it seems that all of the methods in question were able to find a relatively sufficient value that nears the average tempo of the song.

Method[1]	MSE[2]	Error rate	Insignificant errors	Half-BPM detections
BLSTM	13.813	70/773 (9.06%)	198	2
Complex	17.344	48/773 (6.21%)	124	1
CNN	8.424	56/773 (7.24%)	197	3
Energy	54.041	129/773 (16.7%)	148	10
HFC	44.041	68/773 (8.80%)	120	2
LL (RNN)	14.153	40/773 (5.17%)	152	0
MKL	16.176	55/773 (7.12%)	115	2
KL	27.468	77/773 (9.96%)	140	2
PB	262.734	288/773 (37.3%)	158	33
SD	36.647	61/773 (7.89%)	135	6
SF	33.291	47/773 (6.08%)	123	1
`aubio`	638.627	772/772 (100%)	0	130
`madmom`	151.536	686/772 (88.9%)	3	381
Spotify	116.462	388/772 (50.3%)	376	264

[1] Italicized methods are neural-network-based.
[2] The mode is used as ground truth.
[3] Highlighted cells indicate the best value in a column. Insignificant errors are not highlighted because the best candidates in the column received the most significant errors.

Table 3: Comparison of various onset and tempo detection methods on the Spotify dataset.

Table 3 shows the various onset and tempo detection methods tested on the Spotify dataset. BPM values within ±0.05 of the "ground truth" values are not considered errors, as such a small difference would not be noticeable unless working with music that spans hours of time, allowing enough time for the audio to desync. The amounts of insignificant errors are also detailed on the table. The last column notes the half-BPM detections; for most music in common time, also known as a 4/4 time signature, halving or doubling a BPM value still follows the same rhythm and for our purposes are treated and calculated as equivalent; however, if this is a problem it would be best to avoid algorithms with high amounts of half-BPM detections.

The first observation of note is that the machine learning-based methods all had the lowest mean squared error, suggesting that they may be notably more precise than the other methods. The CNN method had the lowest error of about 8.4, which is significantly smaller than the others. In terms of the significant error count, the best methods were Complex Domain, Spectral Flux and LL, with LL having the lowest error rate of about

5%. This suggests that LL may be a fairly robust option along with the other methods. Machine learning-based methods were more likely to receive insignificant errors with slight deviations in the tempo, especially with BLSTM and CNN; in most cases this can be mitigated by rounding to the nearest 0.05. In terms of half-BPM detections, the majority of methods barely had any, but LL was the only method with zero half-BPM detections.

When using van de Wetering's method, Energy-Based Distance and Phase-Based were considerably worse than the other onset detection methods; Energy-Based Distance received significantly more errors and half-BPM detections than the other algorithms, and Phase-Based had the worst mean squared error by far with over 260.

Observing the other tempo detection methods, `aubio` never reached below 0.05 error, often calculating values within 20 BPM and sometimes within 2 BPM of the correct values, but never reaching the ground truth value. `madmom`'s tempo detection algorithm showed a high error rate but also had limited insignificant errors; when it calculated a correct value it generally was the exact same. `madmom` also tended to detect half the BPM of the ground truth very frequently compared to every other model. Spotify's tempo detection underperformed significantly compared to any onset method used in tandem with van de Wetering's implementation, but still managed to exceed aubio and madmom's accuracy and precision; however, it did also receive a considerable amount of insignificant errors and half-BPM detections.

5 Analysis

Notably, none of the other tempo detection methods succeded in reaching comparability to van de Wetering's implementation. The overwhelming difference in error rate shows how important it is to introduce a working tempo detection system and open it for public use so that it is easier to find the correct values without doing the process by hand. Tempo detection for `madmom` and `aubio` was also notably slow; it took approximately an hour to process all the songs (1.79 seconds per 30-second sample on average for `madmom` and `aubio` combined) while it took about 20 minutes to process the onsets and tempo for all 12 different onset methods using van de Wetering's method. In terms of performance, van de Wetering's method comparatively worked quickly.

For precision, it seems that all of the neural-network-based onset detection algorithms are the best when detecting tempo. However, very slight differences in error are common and rounding is recommended to ensure that they work well. In terms of reliability and small error rates, the Complex Domain, Spectral Flux and LL algorithms shined.

In terms of performance, machine-learning-based methods are naturally slower and more resource-intensive, though all of them work faster than real-time on most devices. Ultimately, Complex Domain and Spectral Flux seem to stand out as the best non-machine-learning based algorithms, and from preliminary testing it would seem that Complex Domain is more precise in its estimations, making that and the LL algorithm the most well-rounded algorithms out of the bunch, especially since both of them can be used in real-time as well.

6 Extensions

From looking at the previous data, it is clear that the current state-of-the-art tempo estimation is not perfect. Ideally the error rate should be as close to 0% as possible, as projects which require synchronization to music need accurate tempo values. The next steps would be to pinpoint the weaknesses of each of the methods and create a large dataset with lots of variety as well as human-confirmed BPM values to reference. Using data from all sorts of musical genres would be ideal. Tempo detection is hardest when looking at music with less defined onsets such as softer jazz, classical music, music without drums, etc. With a better dataset that encompasses a variety of potential edge cases as well as contains more accurate ground truths, it would be more sound to confirm that van de Wetering's implementation exceeds the state of the art in precision for songs with a fixed tempo. It is extremely useful to me and many others to be able to synchronize videos, games, and various media automatically to music without having to go through lengthy trial-and-error processes.

There are other factors than precision and accuracy when evaluating the viability of onset detection methods; it would be interesting to directly compare the performance of onset detection methods as well as various tempo detection methods. Figuring out ways to optimize the performance of onset and tempo detection with techniques such as multithreading is always welcome.

One last note of importance is the detection of half or double BPMs. It is important to observe the tendency for tempo estimation methods to choose the correct tempo multiplied or divided by 2 if it is present, as it would affect its reliability in certain use cases. When using automated tempo detection to synchronize something to music, this phenomenon is generally not an issue as a song will still be synchronized assuming that it is in common time[1]. However, if it is important in research or in practice that a 200 BPM song is detected as faster than a 150 BPM song, then it is important that the correct value is determined. What can make this especially hard is that the idea that a song is in 80 or 160 BPM, etc. is often subjective; many songs could be notated in sheet music either way and song structures could be interpreted differently. It would be very useful to research further into how to solve this issue, and it is possible that an additional layer could be added to the tempo detection process to detect half or double BPM predictions and adjust tempo values accordingly.

7 Implementation

The current implementation is subject to change since I plan to work on this beyond high school, but the following implementation will be up-to-date with the syslab version of this assignment.

The main program file is `FindTempo_standalone.cpp` and includes

[1]It is possible that songs with 3/4 or waltz time signatures may be affected, but it is likely that such music would be miscalculated in multiples of 3 instead. More complex time signatures may cause problems with current tempo detection methods.

FindTempo_standalone.hpp. The following command can be used to compile the program for Linux:

```
g++ FindTempo_standalone.cpp --std=c++17 -laubio -lpthread -lstdc++fs -g -o
FindTempo_standalone
```

The c++17 standard is required for filesystem access during batch processing, otherwise c++11 will work with some removal of code. FindTempo_standalone.hpp includes <experimental/filesystem> which the argument -lstdc++fs refers to; for newer compilers include <filesystem> instead and omit the -lstdc++fs argument.

The code *should* compile on Windows, but I have not tested it. Some lines may need to be changed to allow Visual Studio's compiler to work (specifically, I believe that some functions may need to be converted to their "safe" equivalents).

The required libraries are aubio and, if using PolyfitBoost.hpp (explained below), boost. Batch processing can technically be done without aubio but will require removing code related to onset detection and sample loading.

Batch processing requires a specific format in text files, and the format is as such:

```
Artist
Title
Sample rate (44100 Hz, probably)
Onset detection method
100 (Onsets in samples separated by newlines)
200
300
...
```

It will output a csv file following the schema: artist, title, ID, onset detection method, priority (1st, 2nd, 3rd place), BPM, offset (unused), fitness. Onsets must be calculated prior to processing them in FindTempo_standalone.cpp; I used the aubio and madmom Python libraries to achieve this, and the scripts I wrote are located in the dataset/scripts directory in the repository. Batch processing currently does not support multithreading as there is a memory leak in my threading implementation. This will not affect performance terribly but it can be fixed by replacing the boolean array in the ParallelThreads class with an array of thread pointers and joining them after completion.

The code is able to use two polyfit implementations: PolyfitBoost.hpp and polyfit.h. polyfit.h is the original polyfit implementation used by van de Wetering and is recommended as it seems to work best. PolyfitBoost.hpp is an implementation I used before receiving polyfit.h which seems to yield different results; although the results are usable I have not been able to test why each version results in different code. As such, both versions will be included, and in FindTempo_standalone.cpp the polyfit implementation can be switched by using mathalgo::polyfit() for polyfit.h and polyfit() for PolyfitBoost.hpp.

Originally this project was also supposed to be able to determine the offset in milliseconds of the first beat, making it easy to automatically synchronize anything with a given audio file. However, the offset values seem to be very wrong; in the future I plan to correct this, but as of now the offset values seem to be useless. This may be a result of using a newer version of `aubio` to determine onsets, but I am not sure what the cause is as of present.

For archival (and amusement) purposes, the informal journal entries that I wrote for my research project are included in the repository, and any technical issues and solutions and may be covered there. The original presentation I wrote for school is also included. The original iterations of my code as well as van de Wetering's original `FindTempo.cpp` will be included in the `legacy` folder.

Acknowledgments This project was conducted through the TJHSST Computer Systems Lab as a high school senior research project. I thank Dr. White, my mentor for this research project, for the support and encouragement he has provided throughout the school year. I would also like to thank Bram van de Wetering for being kind enough to share lots of helpful information, code, and feedback regarding musical tempo estimation. Finally, I would like to thank my friend and fellow senior Stephen Huan for helping me figure out how to use LaTeX so I could write this paper.

References

[1] B. van de Wetering, *Non-causal beat tracking for rhythm games*, Mar. 2016.

[2] S. Böck, *Onset, beat, and tempo detection with artificial neural networks*, Diploma thesis. [Online]. Available: http://mir.minimoog.org/sb-diploma-thesis.

[3] S. Böck, A. Arzt, F. Krebs, and M. Schedl, "Online real-time onset detection with recurrent neural networks," in *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx)*, 2012. [Online]. Available: https://www.dafx12.york.ac.uk/papers/dafx12_submission_4.pdf.

[4] J. Schlüter and S. Böck, "Musical onset detection with convolutional neural networks," in *Proceedings of the 6th International Workshop on Machine Learning and Music*, 2013. DOI: 10.1.1.430.9970. [Online]. Available: http://www.ofai.at/~jan.schlueter/pubs/2013_mml.pdf.

[5] S. Böck, F. Krebs, and G. Widmer, "Accurate tempo estimation based on recurrent neural networks and resonating comb filters," in *Proceedings of the 16th International Society for Music Information Retrieval Conference*, 2015. [Online]. Available: http://ismir2015.uma.es/articles/196_Paper.pdf.