Detection of music tempo with beats per minute (BPM) detector

Pedro Macedo
Faculty of Computer and
Information Science
Ljubljana, Slovenia
Email: pf06708@student.uni-lj.si

***Abstract*—The purpose of this project is to classify the tempo of each audio file by checking the audio Bits Per Minute (BPM). The implementation will be done in Python by using open source libraries specified for sound analysis.**

***Index Terms*—Virtual Reality (*VR*), Augmented Reality (*AR*), Spatial Audio, Location-Guided Audio-Visual Spatial Audio Separation (*LAVSS*), Machine Learning (*ML*), Bits-per-minute (*BPM*)**

I. Goals

The main goal of this project is to develop a sophisticated web application that accurately recognises the tempo of music tracks by analysing their *Beats Per Minute* (*BPM*). This functionality is crucial for a variety of practical applications, from music recommendation systems to fitness applications that use the tempo of music to increase user engagement.

The project employs advanced signal processing techniques and leverages the power of open source sound analysis libraries to ensure high accuracy in BPM detection. By integrating these technologies, the system aims to provide precise BPM values for a variety of music genres, which are essential for correctly determining the tempo of the music.

The aim is to provide a robust and user-friendly tool that:

- Accurately recognise and display the BPM of any music track.
- The BPM data is used to classify the music tempo from a predefined BPM tempo list.

II. Progress

The first decision that had to be made concerned the system architecture.

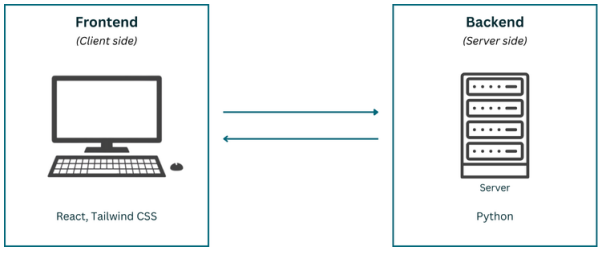

Fig. 1. System Architecture. This figure shows the various components of this system, client and server components.

As we can see in Fig. 1, this system consists of two different components, *backend* and *frontend* components. The backend of the system was developed in *Python* and uses the *FastAPI* library to build an API for our system. The frontend of the system was developed in *React* and uses the *Tailwind CSS* framework for styling our website. It also uses, the *PyDub* library to convert an *MP3* file into a *WAV* file, and the *Librosa* library to load the received audio data and convert it into a floating point time series.

In Fig. 3, you can see the main page of the system, where the user can upload any music file and see the *BPM* values for every second of the song while the song is playing in the background. In addition, the user can view their history by clicking on the button in the top right corner.

Fig. 2. Main page. This figure shows the main page of our system with the various functionalities available.

In order to obtain the overall *BPM* of the uploaded song, the following procedure was established:

1) **File conversion**: Firstly, the file was converted from an *MP3* file type into a *WAV* file type.
2) **Audio division**: Subsequently, the audio was splitted into 1 second samples. By splitting the audio into 1-second samples, it allows to process the audio and detect the music tempo more accurately.
3) **Smooth audio sample**: For each segment, the audio segment was smooth by calculating the its energy, to ensure that the *BPM* detection is based on genuine patterns rather than spontaneous artifacts or noise.

4) **Peaks detection**: Moreover, we detected audio peaks (by defining a peak height threshold and a minimum distance between the detected peaks).
5) **BPM calculation**: If there are sufficient number of peaks, the BPM is calculated based on the average interval between peaks.

$$BPM = f * 60$$
$$BPM = \frac{60}{T}$$

6) **Overall song BPM**: Furthermore, we applied the *DBSCAN* clustering algorithm to determine the overall BPM of a song based on the BPM values detected for each 1-second samples (obtained from the previous step). This clustering algorithm was used since it handles noise and outliers in the audio data. This clustering algorithm forms clusters based on the data density, which helps to detect the most dominant cluster in the song. And, subsequently, to detect the song's BPM by calculating the mean of the BPM values of the most dominant cluster.

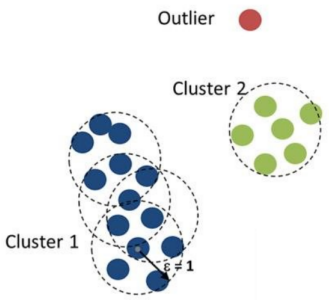

Fig. 3. *DBSCAN* algorithm. Divides the values into clusters and removes outliers.

7) **Detect song *tempo***: After detecting the song *BPM*, the song tempo was determine by a predefined list of *BPM* ranges.

TABLE I
MUSICAL TEMPO RANGES BASED ON BPM

Tempo Category	BPM Range
Very Slow (Below Largo)	0-39
Largo (Very Slow)	40-60
Adagio (Slow and Stately)	61-76
Andante (Walking Pace)	77-108
Moderato (Moderate)	109-120
Allegro (Fast and Lively)	121-156
Vivace (Lively and Fast)	157-176
Presto (Very Fast)	177-200
Prestissimo (Extremely Fast)	>201

III. CHANGES TO THE INITIAL PLAN

The work done so far is accordingly to the plan defined in the initial plan. Despite this, we decided to add a BPM overtime graph that changes while the song is played on the website background. This change made the website more interactive and engaging for users. By visualizing the BPM fluctuations in real-time, users can better understand the dynamics of the music as they listen. This allows users to see how BPM values changes throughout the song.

IV. FUTURE WORK

In order to improve the BPM detection pipeline, I want to use the approach described in the paper "Survey paper on music beat tracking" by *Makarand Velankar* [1]. In this approach, they divided the frequency spectrum into 3 ranges (*Low-frequency band*, *Middle-frequency band* and *High-frequency band*, while using onset and transient detectors to determine the song's *Periodicity Detection Function* (*PeDF*) and then the get the overall song's BPM. This approach will allow for more precise and reliable beat tracking and tempo detection in songs.

V. CONCLUSION

This project demonstrates the development and implementation of a sophisticated system that accurately determines the beats per minute (*BPM*) of the uploaded song.

The integration of the *BPM* overtime graph significantly enhanced the user interface, offering a dynamic and interactive experience.

In conclusion, the current state of this project meets the initial proposed plan. The future work described will be done until the end of the project development phase, and some user interface will be enhanced to improve user's experience on our system.

REFERENCES

[1] M. Velankar, "Survey paper on music beat tracking," *http://www.ijrcct.org/index.php/ojs/article/view/373/pdf*, vol. 2, p. 953, 10 2013.