Assignment 1: BeatRoot

Dekun Xie | xiedekun@outlook.com

1.Introduction

Beat tracking derives from a music audio signal a sequence of beat instants that might correspond to when a human listener would tap his foot [1]. This assignment explores a classic beat tracking method called BeatRoot proposed by Simon Dixon in 2001[2]. This method works without any prior knowledge of the tempo, meter or musical style is assumed; all required information is derived from the data.

2.Datasets

The dataset to use is called 'Ballroom Dataset' which includes beat and bar annotations of the, introduced by Gouyon et al.[3]. It does not include the audio files. The audio files (698 files, size 1.5 GB) can be downloaded from http://mtg.upf.edu/ismir2004/contest/tempoContest/data1.tar.gz.

In the next experiment section, I will choose 'Albums-Ballroom_Classics4-14.wav' which is a 30s song as the example.

3.Implementation

I read the whole paper in 2001 to learn how to implement BeatRoot by myself, and there are two processes in this algorithm. One is 'Tempo Induction' including Onset Detection and Clustering and another is 'Beat Tracking' which contains the procedure of Agents.

3.1 Tempo Induction

In the tempo induction stage, it calculates the times between pairs of note onsets and uses a clustering algorithm to find significant clusters of inter-onset intervals. The interval in the cluster can be regarded as the hypothetical tempo. In the end, the cluster will be ranked and the list of ranked tempo hypotheses will be input in the next stage (Beat Tracking) to be calculated about the beat times. The ranking rule is that the cluster with the most salient time intervals will be ranked most highly.

3.1.1 Onset Detection: Spectral Flux

The Onset Detection stage is to detect the onset between the notes. As the intervals between each onset could be the subset of the beat, it is a fundamental stage for beat calculation.

The onset detection method the 2001 paper referred to is based loosely on the techniques of Schloss (1985)[4] and sets a window of 20ms with a 50% overlap. However, I refer to the onset method of Simon Dixon (2006)[5]. He presented several methods including Spectral Flux, Phase Deviation, Complex Domain and etc. Finally, I choose Spectral Flux to detect onsets. There are two reasons for this decision: firstly, it will be novel to use a different method from a certain part of the original paper and another reason is that, it was said spectral flux has the advantage of being the simplest and is the fastest algorithm among algorithms above.

Spectral flux is a time domain method and detects the changes in the spectrum. It measures the change in magnitude in each frequency bin, and if this is restricted to the positive changes and summed across all frequency bins. This function is defined as:

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

where $H(x) = \frac{x+|x|}{2} = \max(x,0)$ is the half-wave rectifier function.

As for the parameters, I choose the hamming window and set the window time 40ms and the hop time 10ms, which is similar to the parameter in 2001 paper.

Finally, the onset (black lines) is produced as Figure 1.

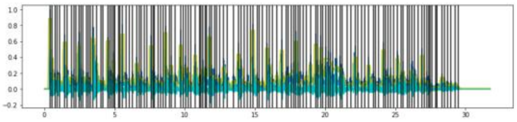
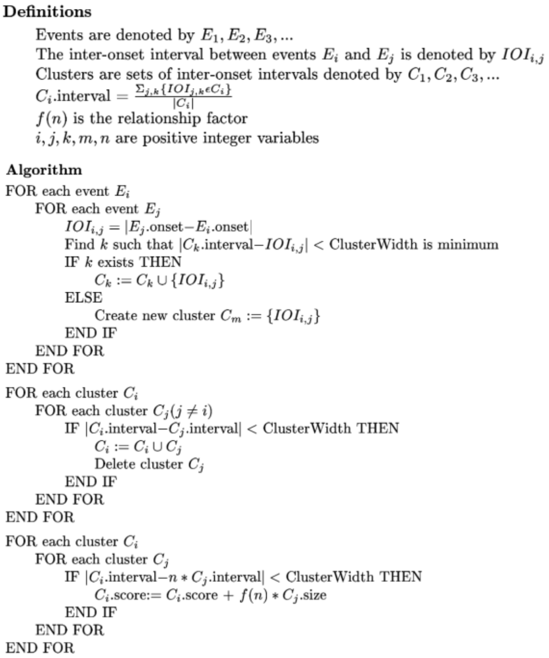


Figure 1. Onset Detection in Albums-Ballroom_Classics4-14.wav

3.1.2 Clustering

The clustering procedure will produce a ranked list of tempo hypotheses by using the onset data calculated in the last section. I will implement the cluster algorithm according to the Figure 2



Definitions

Events are denoted by $E_1, E_2, E_3, \ldots$
The inter-onset interval between events E_i and E_j is denoted by $IOI_{i,j}$
Clusters are sets of inter-onset intervals denoted by $C_1, C_2, C_3, \ldots$
C_i.interval $= \frac{\Sigma_{j,k}\{IOI_{j,k} \epsilon C_i\}}{|C_i|}$
$f(n)$ is the relationship factor
i, j, k, m, n are positive integer variables

Algorithm

```
FOR each event $E_i$
    FOR each event $E_j$
        $IOI_{i,j} = |E_j.\text{onset} - E_i.\text{onset}|$
        Find $k$ such that $|C_k.\text{interval} - IOI_{i,j}| <$ ClusterWidth is minimum
        IF $k$ exists THEN
            $C_k := C_k \cup \{IOI_{i,j}\}$
        ELSE
            Create new cluster $C_m := \{IOI_{i,j}\}$
        END IF
    END FOR
END FOR
FOR each cluster $C_i$
    FOR each cluster $C_j (j \neq i)$
        IF $|C_i.\text{interval} - C_j.\text{interval}| <$ ClusterWidth THEN
            $C_i := C_i \cup C_j$
            Delete cluster $C_j$
        END IF
    END FOR
END FOR
FOR each cluster $C_i$
    FOR each cluster $C_j$
        IF $|C_i.\text{interval} - n * C_j.\text{interval}| <$ ClusterWidth THEN
            $C_i.\text{score} := C_i.\text{score} + f(n) * C_j.\text{size}$
        END IF
    END FOR
END FOR
```

Figure 2. Cluster Algorithm

In the first loop of the algorithm, it calculates the intervals between each onset, which is called IOI (inter onset interval). If the IOI is within a certain value (Cluster Width = 25 ms) of the interval of the cluster, the IOI will be input the cluster and the interval of the cluster should be updated by averaging. Otherwise, if there is no cluster meets the condition, it will create a new cluster to contain the IOI. In a word, this part is to calculate IOI to initialise the cluster.

As for the second loop, the algorithm combines the clusters with similar intervals (within 25 ms) to reduce the number of clusters. This process will delete the redundant cluster information to increase the efficiency in the next ranking stage and beat tracking stage.

The last loop is to rank the cluster in order to produce hypothesis tempi. As the more elements the cluster contains, the higher possibility it would get to be the final beat, the ranking score depends on the number of elements the cluster contains (size), which means the larger the size is, the higher score the cluster gets. Furthermore, if the cluster contains more subsets of clusters with integer relationship intervals, the cluster will be more likely to represent the beat. Therefore, the score of the cluster will be added the size of the subsets multiplying the relationship factor f(d) which is defined as:

$$f(d) = \begin{cases} 6-d, & 1 \le d \le 4 \\ 1, & 5 \le d \le 8 \\ 0, & \text{otherwise} \end{cases}$$

In the end, we get a list of tempo hypotheses and I select 10 top ranked clusters which will represent a set of hypotheses as to the basic tempo of the music (shown in Table 1). However, these sets of tempi are only the probable tempo of the music. We also should calculate the beat in the time sequency in the Beat Tracking stage, especially for those whose tempo is unstable.

Table 1. Best 10 Clusters Detail

Cluster Index	**Interval**	**Size**	**Score**
0	1.7003	37	447
1	1.0104	42	446
2	3.0308	24	446
3	2.0249	33	438
4	3.7156	25	432
5	5.1818	34	431
6	6.8192	24	427
7	1.2398	31	424

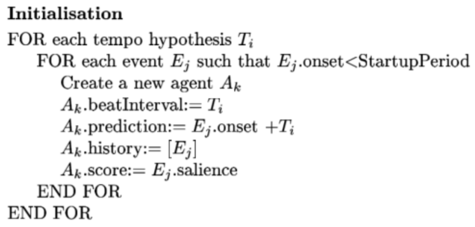
8	6.0781	21	422
9	4.0368	28	421

3.2 Beat Tracking

In this part, I will make use of the tempo hypotheses produced in last stage and generate the agents to predict the next beat. In addition, the tolerant windows will be set to adjust the beat over time.

3.2.1 Agents

Agents is used to predict and produce the final beats over time. The method to initialise the agent is shown below:

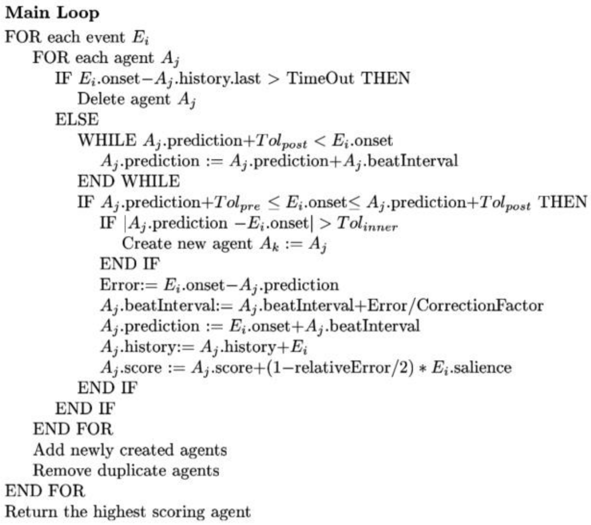


```
Initialisation
FOR each tempo hypothesis $T_i$
    FOR each event $E_j$ such that $E_j$.onset<StartupPeriod
        Create a new agent $A_k$
        $A_k$.beatInterval:= $T_i$
        $A_k$.prediction:= $E_j$.onset +$T_i$
        $A_k$.history:= [$E_j$]
        $A_k$.score:= $E_j$.salience
    END FOR
END FOR
```

Figure 3. Agent Intialisation Algorithm

As we can see, the interval in the cluster will be used as the initial hypothesis beat interval of the agent. And then, we choose each onset as the start of the beat. Because we don't want the initial beat to start in the late time, we only choose the onsets before the Startup Period (5s in the paper). After that, the agents can be initialised: The beat interval is the interval in the cluster; the prediction of next beat is the start onset plus beat interval; the history array contains the start onset and will add more onsets later; the score is the salience (magnitude) of the onset.

After finishing initialisation, the agent will start predict and adjust the beats. The expanded history array will become the final beats we track. The main loop is presented below.



```
Main Loop
FOR each event $E_i$
    FOR each agent $A_j$
        IF $E_i$.onset$-A_j$.history.last > TimeOut THEN
            Delete agent $A_j$
        ELSE
            WHILE $A_j$.prediction+$Tol_{post}$ < $E_i$.onset
                $A_j$.prediction := $A_j$.prediction+$A_j$.beatInterval
            END WHILE
            IF $A_j$.prediction+$Tol_{pre}$ $\leq$ $E_i$.onset$\leq$ $A_j$.prediction+$Tol_{post}$ THEN
                IF |$A_j$.prediction $-E_i$.onset| > $Tol_{inner}$
                    Create new agent $A_k$ := $A_j$
                END IF
                Error:= $E_i$.onset$-A_j$.prediction
                $A_j$.beatInterval:= $A_j$.beatInterval+Error/CorrectionFactor
                $A_j$.prediction := $E_i$.onset+$A_j$.beatInterval
                $A_j$.history:= $A_j$.history+$E_i$
                $A_j$.score := $A_j$.score+(1$-$relativeError/2) $*$ $E_i$.salience
            END IF
        END IF
    END FOR
    Add newly created agents
    Remove duplicate agents
END FOR
Return the highest scoring agent
```

Figure 4. Agent Main Loop Algorithm

As we want the beat close and tight, we just abandon the agents whose last beat is too far from the next onset (far means over 2s in my program). And then, the agent starts to predict the next beat.

Firstly, we will set 3 tolerant window value. We will set the inner window 40 ms either side of the predicted beat time. It represents the deviations from strict metrical time which an agent is willing to accept. In addition, the outer window, with a size of 10% and 20% of the inter-beat interval respectively before and after the predicted beat time in my program, represents changes in tempo or phase which an agent will accept as a possibility. If the beat of the audio is stable, we can set the values of out window smaller. After that, if the onset is still behind the prediction minus the previous tolerance, we continue to compare difference between the prediction and the onset to see whether it larger the inner tolerance. If so, a new agent will be created, which is a way to adjust beats and tempo. Finally, the parameter of the agent will be updated according to the error which is the difference of the prediction of the true onset and then we remove the duplicate agents to increase efficiency before returning the 5 highest scoring agents (Table 2). The beat we track is the onset in the history array in the best scoring agent.

Table 2. Best 5 Agents Detail

Agent Index	**Beat Interval**	**History Length**	**Score**
0	1.0017	26	14.9421
1	1.0259	27	13.3696
2	1.0017	23	12.799
3	1.0009	25	11.7727
4	1.0006	23	11.315

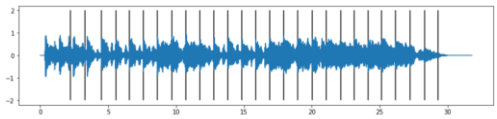


Figure 5. The Best Precision Agent Beats

4.Evaluation

The beat intervals the program calculates may be the integer multiple of the ground truth. For example, if the tempo we calculate is 60 bpm and each beat is the at the true beat although the ground truth is 120 bpm, we can still regard the result totally correct. In this case, the recall is 50% and precision is 100%. So, I will choose precision as an evaluation value. Sometimes it's the other way around: the ground truth will be the integer multiple of the calculated beat intervals where we cannot use precision as evaluation. However, in this program and datasets, the calculated beat intervals will always be larger than the ground truth which means the selecting precision is a reasonable decision.

The precision is defined as:

$$Precision = \frac{TP}{TP + FP}$$

It means how many beats we calculate is correct compared with the ground truth.

I choose one example song from each genre and use the mir_eval package to calculate precision after 5s. The result is shown in Table 3 in Appendix A.

According to the Table 3, we can find that the model works well when the song gets strong accent, for example with strong drum or with strong bass. It is worth noticing that 'Albums-Ballroom_Classics4-14', 'Albums-Cafe_Paradiso-05' and 'Albums-Cafe_Paradiso-14' all get 100% precision. However, the reason why 'Albums-AnaBelen_Veneo-13' gets 0% is that this song has many syncopations which cheats the model to calculate beats after slightly delay. Therefore, we can find that the overall plot of this song is correct in spite of a little delay.

5.Conclusion

The BeatRoot implemented by myself works well when the song has obvious accents, although it doesn't for songs without strong accents. Thus, there might be some aspects I could improve to increase the score of the program.

Firstly, I could improve the onset detection method by trying using other methods such like Complex Domain. As the beat tracking progress is based on the onsets we calculate in the detection step, the improvement of the onset detection may increase the whole consequence.

Secondly, we can set the salience all to 1 to improve the prediction in the agent step. In that step, the larger salience the onset gets, the higher score the agent will get, which means the onset of strong accents will be more likely to be the beat. However, it is not true for that music with plenty of syncopations (e.g., 'Albums-AnaBelen_Veneo-13' which got 0% precision). Therefore, to set every salience value to 1 could eliminate this fake hypothesis. To prove it, I apply this improvement for 'Albums-AnaBelen_Veneo-13' example and the precision increases from 0% to 88.46% shown in Figure 6.

It means this improvement works.

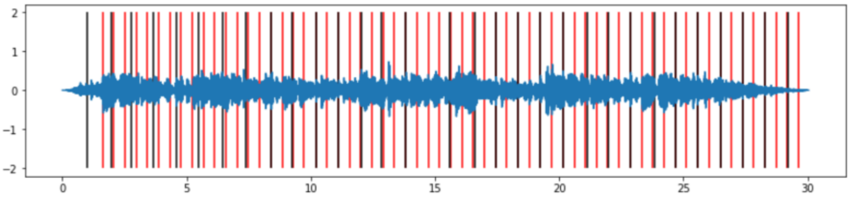


Figure 6. The Improvement in Albums-AnaBelen_Veneo-13

In a word, BeatRoot is excellent method even if it was presented 20 years ago and we can improve that to track the beats more accurately.

Reference

[1] Ellis, D. P. (2007). Beat tracking by dynamic programming. Journal of New Music Research, 36(1), 51-60.

[2] Dixon, S. (2001). Automatic extraction of tempo and beat from expressive performances. International Journal of Phytoremediation, 21(1), 39–58.

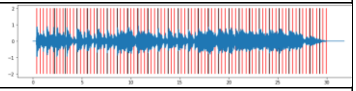
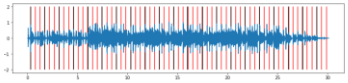
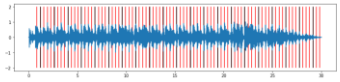
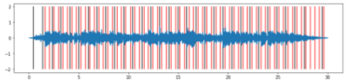
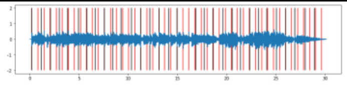
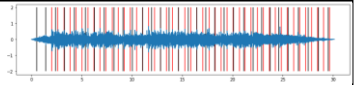
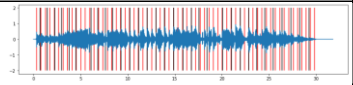
[3] Gouyon F., A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano. An experimental comparison of audio tempo induction algorithms. Transactions on Audio, Speech and Language Processing 14(5), pp.1832-1844, 2006.

[4] Schloss, W. A. (1985). On the Automatic Transcription of Percussive Music--From Acoustic Signal to High-level Analysis (Doctoral dissertation, Stanford University).

[5] Dixon, S. (2006, September). Onset detection revisited. In Proceedings of the 9th International Conference on Digital Audio Effects (Vol. 120, pp. 133-137).

Appendix A

Table 3. Precision Results for 8 Genre Song Datasets

Audio Name	**Genre**	**Strong Drum**	**Strong Bass**	**Precision**	**Plot**
Albums-Ballroom_Classics4-14	Viennese Waltz	False	True	100%	
Albums-Cafe_Paradiso-05	Cha Cha Cha	True	True	100%	
Albums-Cafe_Paradiso-14	Jive	True	True	100%	
Albums-AnaBelen_Veneo-13	Rumba-American	False	False	0%	
Albums-AnaBelen_Veneo-01	Rumba-International	False	False	44%	
Albums-AnaBelen_Veneo-02	Samba	True	False	33.33%	
Albums-Ballroom_Classics4-06	Tango	False	True	65.38%	
Albums-Ballroom_Classics4-01	Waltz	False	False	25.81%