IBT: A REAL-TIME TEMPO AND BEAT TRACKING SYSTEM

João Lobato Oliveira[1,2] **Fabien Gouyon**[1] **Luis Gustavo Martins**[3] **Luis Paulo Reis**[2]
[1]Institute for Systems and Computer Engineering of Porto (INESC Porto), Porto, Portugal
[2]Artificial Intelligence and Computer Science Laboratory (LIACC), FEUP, Porto, Portugal
[3]Research Center for Science and Technology in Art (CITAR), UCP, Porto, Portugal
{jmso,fgouyon}@inescporto.pt lgustavomartins@gmail.com lpreis@fe.up.pt

ABSTRACT

This paper describes a tempo induction and beat tracking system based on the efficient strategy (initially introduced in the BeatRoot system [Dixon S., "Automatic extraction of tempo and beat from expressive performances." Journal of New Music Research, 30(1):39-58, 2001]) of competing agents processing musical input sequentially and considering parallel hypotheses regarding tempo and beats. In this paper, we propose to extend this strategy to the *causal* processing of *continuous* input data. The main reasons for this are threefold: providing more robustness to potentially noisy input data, permitting the parallel consideration of a number of low-level frame-based features as input, and opening the way to real-time uses of the system (as e.g. for a mobile robotic platform).

The system is implemented in C++, permitting faster than real-time processing of audio data. It is integrated in the MARSYAS framework, and is therefore available under GPL for users and/or researchers.

Detailed evaluation of the causal and non-causal versions of the system on common benchmark datasets show performances reaching those of state-of-the-art beat trackers. We propose a series of lines for future work based on careful analysis of the results.

1. INTRODUCTION

Computational tracking of musical beats from audio signal is a very important feature to automated music analysis. In the context of Music Information Retrieval applications, such as e.g. automatic genre classification, music similarity computation, *autotagging*, or query-by-example, recent literature indicates that audio descriptors of higher level of abstraction are needed [1]. It is a relatively safe bet to say that reliable beat trackers will be helpful in this endeavour.

Recent evaluations of existing beat tracking systems (see e.g. MIREX [1]) show that, although progresses have undeniably been achieved in the last years, there is still room for improvement. Many open directions to beat tracking research are also detailed in a recent and very thorough evaluation in [6]. Particularly, there is to date, to our knowledge, no real-time and open-source audio beat tracker available.

Very many papers in the literature address the problem of tempo induction and beat tracking of audio signals. Providing a review of existing systems and algorithms is out of the scope of this paper. Interested readers are referred to [8] for a review of rhythm description systems.

It is however important to mention the main functional aspects commonly found in beat tracking algorithms. A generic description includes the following computing blocks: (1) Audio feature extraction, (2) Induction (or "Pretracking" herein), and (3) Beat Tracking *per se*.

It is also interesting to notice that recent systems, e.g. [12], [7], [3], [14], tend to implement beat tracking as a repeated induction process, in which tempo and beats are computed on consecutive windows of signal (usually a few seconds, where it is usually considered that the tempo is constant), with overlap, and in which estimating tempo evolution and beat positions is done by connecting observations between windows. We argue that a problem with this approach is a potential computational overload, the intrinsic difficulty to adapt these tracking strategies to causal and real-time scenarios, as well as lack of continuity between windows. Instead, we propose to follow the tracking strategy initially proposed in the system BeatRoot [4], where competing agents process musical input data sequentially and consider parallel hypotheses regarding tempo and beats.

We propose to differ from BeatRoot's strategy by implementing a *causal* decision process over competing agents (instead of taking decisions after the whole data has been analysed). Further, we extend the algorithm to the processing of *continuous* input data. Our aim is to provide more robustness to potentially noisy input data, and opening the way to (faster than) real-time uses of the system (as e.g. for a mobile robotic platform). The system is implemented in C++ and the source code is available as GPL. Although this paper does not provide experiments with respect to the usefulness of diverse low-level features as input to tracking beats [9] [2], it should be noted that a particularity of the proposed architecture is precisely to be open to such experiments. Another difference with BeatRoot lies in an attempt to not bias results towards faster metrical levels.

[1] http://www.music-ir.org/mirex/2009/

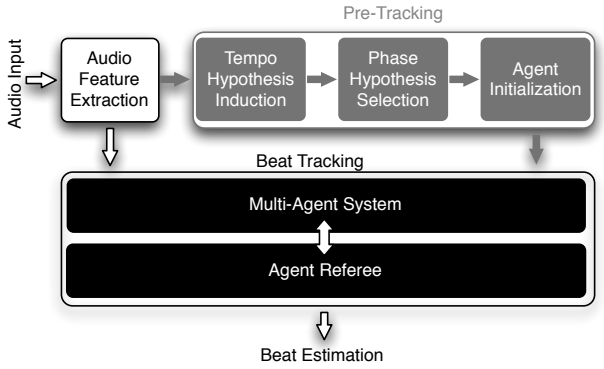

Figure 1. IBT block diagram.

In section 2, we describe one-by-one the functional blocks of IBT, [2] a tempo induction and beat tracking algorithm in the line of BeatRoot [4]. The algorithm follows a modular workflow composed by: (1) an audio feature extraction module, "parsing" the audio data into a continuous [3] feature sequence assumed to convey the predominant information relevant to rhythmic analysis; followed by (2) a pre-tracking module, which outputs initial hypotheses regarding possible beat periods and phases; followed by (3) a beat tracking module, which propagates hypotheses, proceeds to their online creation, killing and ranking, and outputs beats on-the-fly (see Figure 1). Section 3 reports on a thorough evaluation of the system. Section 4 proposes some practical hints for those intending to use the system, and/or make changes to its code. Section 5 discusses the system performances and proposes lines for future work.

2. SYSTEM DESCRIPTION

2.1 Audio Feature Extraction

According to recent comparative studies evaluating alternative onset detection functions [5] and the accuracy of several low-level features applied to beat tracking purposes [9], we selected the spectral flux as the audio feature over which all further processing will be done.

Our implementation follows that proposed in [5]. Particular parameters are: Hamming window, window size of 1024 samples (23.2ms at a sampling rate of $F_s = 44100Hz$), and 50% overlap.

In order to smooth the onset detection function and reduce false detections, a low-pass Butterworth filter is applied on the extracted spectral flux values. As a way to avoid phase distortion the spectral flux values in the induction window are filtered in both the forward and reverse directions, resulting in a precisely zero-phase distortion.

2.2 Pre-tracking

The system is initialized on an induction window, set to a length of 5s. The following sections (until Section 2.3) report on computations done on the induction window only. During the processing of that bit of data, the system does not output beats. At the end of that pre-processing step, hypotheses regarding periods, phases and scores (P_i, ϕ_i, S_i) of a number of beat agents are passed along to the beat tracking module.

The length of the induction window is a high-level parameter that the user can define.

2.2.1 Period Hypotheses Induction

The first step in the pre-tracking stage is to compute a continuous periodicity function, based on the spectral flux autocorrelation, along time-lags τ :

$$A(\tau) = \sum_{n=0}^{m} SF(n)SF(n+\tau), \qquad (1)$$

where $SF(n)$ is the (smoothed) spectral flux for frame n, and m is the induction window size (in frames).

The periodicity function is then parsed by an adaptive peak-picking algorithm to retrieve N global maxima, whose time-lags constitute the initial set of period hypotheses P_i :

$$\begin{cases} P_i = \arg\max_i(A(\tau)), i = 1, ..., N \\ A(\tau) > \delta * \frac{rms(A(\tau))}{M} \end{cases}, \qquad (2)$$

where δ is a fixed threshold parameter, empirically set to 0.75, and *M* is the chosen tempo range, defined to $[50, 250]$ BPM (i.e. periods of 240 ms to 1.2 s), at a 6ms granularity.

2.2.2 Phase Hypotheses Selection

For each one of the period hypothesis P_i, a number of phase hypotheses ϕ_i^j (where j is the index of the alternative hypotheses for the i-th period hypothesis) are considered among detected onsets (detection is done on the induction window only, and computed as proposed in [5]).

For each period hypothesis, we generate an isochronous sequence of beats (a "beat train template") of constant period for each possible phase ϕ_i, with the same length as the induction window.

Using a simplified tracking procedure (see Section 2.3), considering a constant tempo and phase, we then select the beat train template that best matches the detected onsets and retrieve its corresponding phase [10].

At this point, we have computed a set of period and phase hypotheses, (P_i, ϕ_i). The next step is to compute a score for each hypothesis and to rank them.

2.2.3 Agents Setup

A raw score S_i^{raw} is given to each (P_i, ϕ_i) hypothesis, corresponding to the sum of time deviations between elements of the chosen beat train template and local maximum in the *spectral flux* (see eq. (10)).

Scores are then updated via the consideration of possible metrical relationships between each pair of period hypotheses n_{ij}. As proposed in [4], we define a score S_i^{rel} that favors candidates whose periods are in integer relationships:

$$S_i^{rel} = 10 * S_i^{raw} + \sum_{\substack{j=0 \\ j \neq i}}^{N} r(n_{ij}) * S_j^{raw} \qquad (3)$$

[2] Standing for INESC Porto Beat Tracker.

[3] i.e. sampled, with typical sampling rate in the tenth of msec

$$r(n) = \begin{cases} 6-n, 1 \leq n \leq 4 \\ 1, 5 \leq n \leq 8 \\ 0, otherwise \end{cases} \quad (4)$$

Finally, we define the final scores, S_i, as follows:

$$S_i = S_i^{rel} * max(S^{raw}) \quad (5)$$

The estimated hypotheses (P_i, ϕ_i, S_i) can now be used to initialize a set of N beat agents, which will start their beat tracking activity, as described in the following sections.

2.3 Beat Tracking

Following the pre-tracking stage described in the previous sections, the process of on-line beat tracking will consist on the supervision of the incoming spectral flux values, constantly handling any tempo/timing variations, while keeping a good balance between reactiveness (speed of response to system changes) and inertia (stability of the system). As illustrated in Figure 1, this process is handled by a multi-agent system mediated by a central referee.

2.3.1 Agents Operation

Initialized using the pre-tracking (P_i, ϕ_i, S_i) hypotheses, an initial set of N beat agents will start to propagate, in a causal manner, predictions based on incoming data, by representing alternative hypotheses regarding beat positions and tempo. Each prediction is evaluated with respect to its deviation (i.e. error) to the local maximum in the observed data, within a two-level tolerance window. This is a stage where the system differs significantly from BeatRoot: although the tolerance windows are akin to [4], processing continuous data is necessarily different here than onsets; and the generation of new agents also differs as we include more than one new hypothesis, accounting more specifically for tempo and/or timing deviations.

This two-level tolerance window consists in an *inner* tolerance region, $T_{in} \in [T_{in}^l, T_{in}^r]$, $T_{in}^l = T_{in}^r = 46.4$ ms, for handling short period and phase deviations, and an asymetric *outer* tolerance region, $T_{out} \in [T_{out}^l, T_{in}^l[\cup]T_{in}^r, T_{out}^r]$, with a left margin $T_{out}^l = 0.2 * P_i$ and a right margin $T_{out}^r = 0.4 * P_i$, see Figure 2. This allows to contemplate eventual sudden changes in tempo expression (the asymmetry reflects the higher tendency for tempo reductions than increases).

Consequently, two alternative scenarios arise. A first scenario corresponds to a local maximum found inside the *inner* tolerance window. In such case, the agent's period and phase are compensated by a fraction of that error:

$$\begin{cases} P_i = P_i + 0.25 * error \\ \phi^i = \phi^i + P_i + 0.25 * error \end{cases}, \exists\, m \in T_{in}. \quad (6)$$

A second scenario considers bigger deviations, with local maxima in the *outer* tolerance window. On this condition the agent under analysis keeps its period and phase but, in order to cope for potential sudden variations of tempo and/or timing, it generates three children $\{C_1, C_2, C_3\}$ to follow three alternative hypotheses, considering alternative possible deviations of its own current hypothesis: timing (phase), tempo (period), or timing and tempo:

$$C_1 : \begin{cases} P_C^1 = P_i \\ \phi_C^1 = \phi^i + P_i + error \end{cases}, \exists\, m \in T_{out}, \quad (7)$$

$$C_2 : \begin{cases} P_C^2 = P_i + error \\ \phi_C^2 = \phi^i + P_i + error \end{cases}, \exists\, m \in T_{out}, \quad (8)$$

$$C_3 : \begin{cases} P_C^3 = P_i + 0.5 * error \\ \phi_C^3 = \phi^i + P_i + 0.5 * error \end{cases}, \exists\, m \in T_{out}. \quad (9)$$

To keep the competitiveness, these new agents inherit a portion (80% in the current implementation) of their father current score.

Ultimately, alternative possible situations may terminate an agent operation, at any analysis frame: *replacement*, *redundancy*, *obsolescence*, or *loss*. An agent is killed if it is currently the worst agent in a pool of agents that has reached a maximum number (limited to 30 agents), and if its score is lower than a newly created agent. In order to increase the algorithm efficiency, an agent is killed if it is duplicating the work of another agent whose score is bigger (their periods do not differ by more than 11.6ms and their phases no more than 23.2ms). An agent is also terminated if the difference between its score and the best agent's is higher than 80% of the best score. Finally, an agent may be also killed if it seems to be "lost," suggested by a high number (i.e. 8) of consecutive beats predictions outside its inner tolerance window.

2.3.2 Agent Referee

In order to determine the best agent at each data frame, a central *Agent Referee* keeps a running evaluation of all agents at all times. This is conducted by scoring the beat predictions of each agent with respect to its goodness-of-fit to incoming data.

The following evaluation function, Δs, is applied around each beat prediction b_p, which evaluates distance between beat prediction and the local maximum *m* inside either the *inner* or the *outer* window (see Figure 2):

$$\begin{cases} \Delta s = \left(1 - \frac{|error|}{T_{out}^r}\right).\left(\frac{P_i}{P_m}\right).SF(m), \exists\, m \in T_{in} \\ \Delta s = -\left(\frac{|error|}{T_{out}^r}\right).\left(\frac{P_i}{P_m}\right).SF(m), \exists\, m \in T_{out}, \end{cases} \quad (10)$$

where P_m is the maximum admitted period, in frames. The $\frac{P_i}{P_m}$ fraction is used to normalize the score function by the period as a way to deflate faster tempi hypotheses, which would otherwise tend to get higher scores due to a higher number of beat predictions. Note also the fact an agent score can undergo positive as well as negative updates.

2.3.3 Non-Causal Version

Whereas causal processing retrieves the beats of the *current* best agent, at any time-frame, in the non-causal version only the *last* best agent is considered. For such, every agents keep an history of their beat predictions, attached to the one inherited form their relatives, and transmit it to

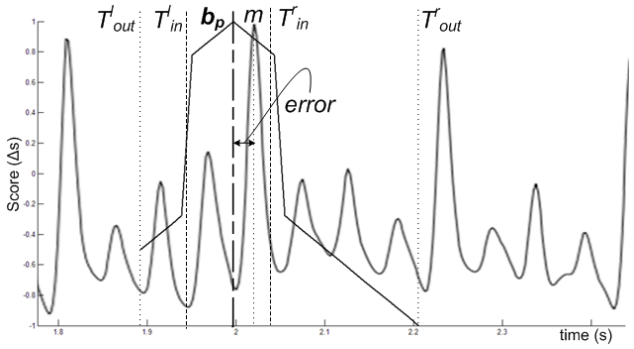

Figure 2. Score function around a beat prediction, b_p, with $P_i = 120BPM$. Example of local maxima m found in the considered inner tolerance window T_{in}.

future generations. Distinctively to the former, this process distinguishes the family of agents whose cumulative score prevails for the whole piece.

In the non-causal version, after pre-tracking and intial agents setup, the analysis "jumps back in time," and beat tracking is performed from the beginning of the signal.

3. EVALUATION

In this section we report on performance evaluation of the proposed algorithm with respect to 2 tasks: tempo estimation and beat tracking. In order to ease comparison to current state-of-the-art systems, we use current benchmark datasets and evaluation measures.

3.1 Datasets

IBT was evaluated using two distinct datasets. For measuring global tempo estimation performance, we use the ISMIR 2004 Tempo Induction Contest data [11]. It consists on 3199 tempo-annotated instances, divided in three categories: *Ballroom*, *Loops*, and *Songs*.

For the beat tracking evaluation, we use 1360 beat-labeled musical pieces (previous use of this dataset is reported in [9] and [6]).

3.2 Evaluation Measures

The system estimation of tempo is evaluated via the two metrics proposed in [11]: *a1* (estimations are considered correct only if they are equal to the annotated tempo) and *a2* (correct estimations also include related metrical levels at 2, 3, $\frac{1}{2}$, and $\frac{1}{3}$ of the ground-truth). Both metrics allow a 4% tolerance window.

Beat-tracking performances are measured via the *P-score* [13], with a 20% tolerance around median Inter-Beat-Interval (IBI) annotations (as in MIREX 2006 Audio Beat Tracking Contest and [6]).

In order to evaluate IBT's robustness to noise distortions, we also applied a number of signal degradations: downsampling, GSM encoding/decoding, fitering, volume adjustment, addition of reverb and white noise (see [11] for more details).

3.3 Global Tempo Estimation

Table 1 presents accuracies obtained for global tempo estimation, with regular and distorted data. The global tempo was measured as the median IBI of final beat predictions, i.e. after beat tracking the whole piece. We also report accuracies obtained before tracking, at the output of the pre-tracking stage, where we select the period hypothesis with highest rank.

Condition	Ballroom		Loops		Songs		Overall	
	a1	a2	a1	a2	a1	a2	a1	a2
IBT(c)	48	83	41	73	30	73	40	76
IBT(c) dist.	44	76	40	72	31	66	38	71
IBT(nc)	49	90	37	76	36	82	41	83
IBT(nc) dist.	48	82	37	74	34	74	40	77
pre-tracking	42	75	40	74	29	71	37	73

Table 1. Global tempo estimation accuracies, in %; "(c)" and "(nc)" stand for the causal and non-causal versions of the system, repectively; "dist." indicates distorted data (see 3.2).

Condition	Metrical relation to annotation (theor. max)					
	1:1(100)	2:1(50)	1:2(50)	3:1(33)	1:3(33)	all
IBT(c)	74(*558*)	47(*282*)	46(*201*)	40(*14*)	27(*9*)	57
IBT(nc)	81(*613*)	46(*266*)	45(*238*)	40(*15*)	24(*10*)	61
1beat(c)	80(*544*)	49(*354*)	40(*109*)	39(*13*)	26(*7*)	59
1beat(nc)	88(*618*)	47(*321*)	42(*153*)	38(*18*)	26(*6*)	64
2beats(c)	79(*1087*)	53(*20*)	28(*2*)	— (*0*)	— (*0*)	72
2beats(nc)	82(*1080*)	47(*44*)	37(*13*)	— (*0*)	— (*0*)	74
dist.(c)	73(*547*)	46(*247*)	45(*164*)	39(*13*)	20(*9*)	55
dist.(nc)	81(*599*)	45(*255*)	44(*196*)	37(*17*)	22(*9*)	59
dind.(c)	72(*511*)	48(*369*)	45(*128*)	38(*22*)	23(*5*)	55
dind.(nc)	81(*582*)	47(*347*)	44(*168*)	37(*15*)	29(*4*)	60
BeatRoot	81(*613*)	48(*535*)	44(*7*)	34(*41*)	N/A	60
BR 2beats	80(*1245*)	45(*10*)	32(*4*)	33(*3*)	N/A	77

Table 2. Beat tracking *P-scores* by metrical relation with the ground-truth, under different conditions; "(c)" and "(nc)" stand for the causal and non-causal versions of the system, repectively; "dist." indicates distorted data (see 3.2); "dind." stands for "dumb" induction (see text). The first line of the table indicates the metrical relation found between the algorithm output and the ground-truth and the corresponding theoretical maximum *P-Score*. The format of other lines is as follows: {P-Score (*number of excerpts tracked at each metrical level)*}.

3.4 Beat Tracking

Table 2 provides results of beat tracking experiments under diverse conditions. The first two lines show results of the causal and non-causal system under regular conditions.

The next four lines refer to beat tracking with biased initialization, either giving the annotated first beat, or giving the first two beats. This help evaluating the performance of beat tracking *per se*, independently of the performance of tempo induction and phase estimation. It is also convenient in order to compare with BeatRoot performances [6].

Results were grouped with respect to metrical relations between the system outputs and the ground-truth annotations. This provides useful information regarding the system tracking performance regardless of it having choosen the "correct" metrical level. (Note that given the evaluation metrics used (the *P-score*), the theoretical performance maximum is different for different metrical relations between output and annotations.)

All results were generated with the same default parameters concerning reactiveness vs. stability of the system. In terms of computational time, IBT took around 11% of the dataset length to process it non-causally, and about 10% to do it causally. (The tests were run on a Core2Duo 2.8 GHz Windows Vista (32-bit) machine.)

4. PRACTICAL USE

IBT was developed in C++ and is freely available, under GPL licensing, in MARSYAS (`http://marsyas.info/`). (At the date of writing, revision 3827.) The algorithm includes three main modes of operation, executable with the following commands:

$ ./ibt input.mp3 (causal mode (default));
$ ./ibt -mic (live mode (microphone captured data));
$ ./ibt -nc input.mp3 (non-causal mode);
$ ./ibt -a input.mp3 (play audio w/ clicks on beats).

4.1 Important parameters

The presented evaluation was run with default parameters, empirically chosen to conciliate reactivity and stability of the system. Values of diverse parameters can be increased to obtain a more reactive system: the margins of tolerance windows (*LFT_OUTTER_MARGIN, RGT_OUTTER_MARGIN, INNER_MARGIN*); portion of an agent current score transmitted to its children, (*CHILDREN_SCORE_FACTOR*); and children correction factors (*CHILDX_FACTOR*).

5. DISCUSSION AND FUTURE WORK

5.1 On tempo estimation

Table 1 shows that the non-causal version of IBT performs comparatively to the best algorithms tested in the ISMIR 2004 contest [11]. The non-causal version shows slightly worse results, but still remains in the best third of the algorithms. Overall it is fair to say that the system finds either the correct tempo or make (somehow acceptable) errors of metrical level in 80% of the cases.

Comparable tempo estimation results are observed on the second dataset (Table 2). Careful evaluation of the first and second lines shows that the tempi of a total of 1064 excerpts and 1142 excerpts (i.e. 78% and 83%) are correct or correspond to metrical level errors, in the causal and non-causal versions, respectively.

The last line of Table 1 also shows that tempo estimation is more reliable after tracking the whole excerpt than at the output of the induction stage. For instance, non-causal beat tracking outperforms pre-tracking by around 10 points. Also, tempo induction seems to work worse on the *Loops* dataset. This is due to the fact that many of these excerpts are very short (many are in fact shorter than our induction window length —5s).

It is also interesting to notice that, after tracking the whole piece, tempo estimation results obtained with "dumb" induction are sensibly similar —albeit an apparent decrease of the number of correct metrical levels found— to those obtained with a more informed induction process (82% vs. 83%, respectively, in the non-causal case, considering all acceptable metrical levels).

Tempo estimation is quite robust to distortions of the audio signal, although the accuracy loss is still about 5 points. This is a clear advantage with respect to systems that process discrete lists of onsets instead of continuous features, such as BeatRoot (see [11] for a detailed comparison — note that BeatRoot's results are relative to a previous version of the software and that recent changes in the onset detection function are likely to have improved them).

These findings seem to indicate that, although tempo induction in IBT reaches good levels, comparable to the state-of-the-art, further increase in accuracy will certainly be obtained if future work is dedicated to improving the induction process. Previous findings indicate that worthwhile lines of work include research on the amount of data needed for induction, reliability of the estimation, improved robustness to noise, and the possibility to trigger induction on different parts of the data, depending on a monitoring of the tracking process self-evaluation.

We can see on Table 1 that accuracy with a2 is much better than with a1 (36-37 points overall difference). Table 2 also shows that, as BeatRoot, IBT tracks a significant number of excerpts at the "wrong" metrical level. However, at the difference with BeatRoot, these excerpts are more uniformally distributed among lower and higher levels. This is the direct effect of the period normalization factor found in the scoring function, eq. (8). These findings indicate that more work should be done on the issue of finding the "correct" metrical level, which may be contemplated by the scoring function itself. In that respect, results from [12] on a1 indicate that a promising direction lies in beat tracking at several metrical levels simultaneously.

5.2 On beat tracking

Table 2 permits us to focus on the tracking performance of the system, independently of its performance in finding the correct tempo. The first two lines of the first column shows us that when IBT finds the correct tempo, it tracks beats correctly in 74% of the cases, the non-causal version does it slightly better: 81%. This is the same performance as BeatRoot. Tracking performances when IBT follows beats on a different metrical level than the annotations are

also similar to BeatRoot.

Careful listening and visualization to tracking errors produced by the causal vs. the non-causal system shows, as should be expected, that the former is more prone to interchanges between phase and metrical levels, compromising continuity.

When one correct beat is given as input to IBT, performance increases up to 88% in the non-causal case. This is a good result, although it also suggests that a number of errors are made at the stage of phase selection (2.2.2) during pre-tracking. Here again, as argued in the previous section, this suggests than future work should be dedicated to improving the induction phase. When two correct beats are given, global performance increases, although performance at the correct metrical level suffers a slight decrease, due to the fact that this figure is computed on significantly more data (i.e. IBT finds more correct metrical levels).

With regards to robustness to signal distortions, it seems that as with tempo estimation, the use of continuous features instead of discrete onsets results in higher robustness. However, IBT performance still decreases about 2 points on average with respect to clean data, calling for future work related to more robust feature extraction.

When the tracking module is given "dumb" period hypotheses, tracking results are only marginally lower than when the period is inferred with a more informed method. This shows that the system has the desirable property to not depend too heavily on correct estimation of the tempo and to recover from errors. Future work should be dedicated to evaluating the speed at which the system recovers from errors, and experiments should be dedicated to fine-tuning system parameters towards the best trade-off between reactiveness to changes and error recovery, on one side, and stability on the other side.

6. SUMMARY

This paper presents IBT, an agent-based tempo and beat tracking system that causally (and non-causally) processes incoming values of a continuous audio feature (e.g. onset detection function). Benchmarks on causal and non-causal versions reveal competitive results, under alternative conditions. In particular, the proposed algorithm produces equivalent beat tracking results to those of BeatRoot, and accurately estimates tempo at the level of state-of-the-art algorithms. A special care has been put on designing a system usable for real-time processing, with good noise robustness, and with no bias towards particular metrical levels. IBT is open-source and freely available with MARSYAS. Promising paths for future work include: tempo induction improvements; informed alternates of the induction and tracking phases; beat tracking at several metrical levels simultaneously; use of several input features.

7. ACKNOWLEDGEMENTS

This work was funded by a PhD scholarship endorsed by FCT, with ref. SFRH/BD/43704/2008, and was supported by QREN's project Palco 3.0, led by Palco Principal.

8. REFERENCES

[1] J.-J. Aucouturier. Sounds like teen spirit: Computational insights into the grounding of everyday musical terms. In J. Minett and W. Wang, editors, *Language, Evolution and the Brain, Frontiers in Linguistics Series*. Taipei: Academia Sinica Press, 2009.

[2] M. Davies and M. Plumbley. Comparing mid-level representations for audio based beat tracking. In *Proceedings of the DMRN Summer Conference*, 2005.

[3] M. Davies and M. Plumbley. Context-dependent beat tracking of musical audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(3):1009–1020, 2007.

[4] S. Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30(1):39–58, 2001.

[5] S. Dixon. Onset detection revisited. In *in Proceedings of the 9th International Conference on Digital Audio Effects*, pages 133–13, Montreal, Canada, 2006.

[6] S. Dixon. Evaluation of the audio beat tracking system beatroot. *Journal of New Music Research*, 36(1):39–50, 2007.

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

[8] F. Gouyon and S. Dixon. A review of automatic rhythm description systems. *Computer Music Journal*, 29(1):34–54, 2005.

[9] F. Gouyon, S. Dixon, and G. Widmer. Evaluating low-level features for beat classification and tracking. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2007.

[10] F. Gouyon, P. Herrera, and P. Cano. Pulse-dependent analyses of percussive music. In *AES 22nd International Conference on Virtual, Synthetic and Entertainment Audio*, 2002.

[11] 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*, 14(5):1832–1844, 2006.

[12] A. Klapuri, A. Eronen, and J. Astola. Analysis of the meter of acoustic musical signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):342–355, 2006.

[13] M. F. McKinney, D. Moelants, M. Davies, and A. Klapuri. Evaluation of audio beat tracking and music tempo. *Journal New Music Research*, 36(1):1–6, 2007.

[14] G. Peeters. Template-based estimation of time-varying tempo. *EURASIP, Journal on Applied Signal Processing (Special Issue on Music Information Retrieval Based on Signal Processing)*, 36(1):51–60, 2007.