An experimental comparison of audio tempo induction algorithms

Fabien Gouyon*, Anssi Klapuri, Simon Dixon,
Miguel Alonso, George Tzanetakis, Christian Uhle, Pedro Cano

***Abstract*— We report on the tempo induction contest organised during the International Conference on Music Information Retrieval (ISMIR 2004) held at the University Pompeu Fabra in Barcelona in October 2004. The goal of this contest was to evaluate some state-of-the-art algorithms in the task of inducing the basic tempo (as a scalar, in beats per minute) from musical audio signals. To our knowledge, this is the first published large scale cross-validation of audio tempo induction algorithms.**

Participants were invited to submit algorithms to the contest organiser, in one of several allowed formats. No training data was provided. A total of 12 entries (representing the work of 7 research teams) were evaluated, 11 of which are reported in this document. Results on the test set of 3199 instances were returned to the participants before they were made public. Anssi Klapuri's algorithm won the contest.

This evaluation shows that tempo induction algorithms can reach over 80% accuracy for music with a constant tempo, if we do not insist on finding a specific metrical level. After the competition, the algorithms and results were analysed in order to discover general lessons for the future development of tempo induction systems. One conclusion is that robust tempo induction entails the processing of frame features rather than that of onset lists. Further, we propose a new "redundant" approach to tempo induction, inspired by knowledge of human perceptual mechanisms, which combines multiple simpler methods using a voting mechanism.

Machine emulation of human tempo induction is still an open issue. Many avenues for future work in audio tempo tracking are highlighted, as for instance the definition of the best rhythmic features and the most appropriate periodicity detection method.

In order to stimulate further research, the contest results, annotations, evaluation software and part of the data are available at `http://ismir2004.ismir.net/ISMIR_Contest.html`

***Index Terms*— Tempo Induction; Evaluation; Benchmark.**

EDICS Category: 2-MUSI

I. Introduction

Much effort in the computer music community has been dedicated to the automation of the beat induction and tracking tasks: determining the basic tempo (rate of musical beats in time, sometimes called the "foot-tapping" rate) and the positions of individual beats in musical files or streams. A number of diverse formalisms have been used to implement computer systems performing these tasks; a survey can be found in [1]. There are many applications of automatic beat induction and tracking of audio signals, e.g., audio-to-score transcription, musical performance research, retrieval of musical pieces with similar tempo to a specific query, generation of playlists, automatic sequencing of musical pieces, determination of boundary points for audio editing operations (cut-and-paste, looping, synchronisation with MIDI clocks or other audio sources), application of beat-synchronous audio effects, visual animations and rhythmic expressiveness transformations.

In any computational modelling endeavour, systematic evaluations play an important part. They require on the one hand reference examples of correct analyses, that is, large and publicly available *annotated data sets* (which in turn calls for an agreement on the manner of representing and annotating relevant information about this data) and on the other hand *agreed evaluation metrics*.

Such evaluations have received little attention in the field of tempo induction and tracking. Early models usually did not present quantitative evaluation of the proposed models, and only recently have researchers begun to report on the performance of their systems, but they meet with the following difficulties:

First of all, even if a number of papers propose evaluation methodologies, no consensus has been reached on how to evaluate algorithms, because of the diversity of input and output data representations as well as the diversity of applications [2]. For instance, Temperley [3] convincingly highlights shortcomings of metrics proposed by Goto and Muraoka [4] and Cemgil et al. [5], and proposes an evaluation method that seems suitable for systems processing MIDI input. However, as this metric is based on a note-by-note evaluation (not beat-by-beat), in order for it to be useful for acoustic signal inputs, it would require complete transcriptions of these signals, an unrealistic requirement from the point of view of manual annotation, and well beyond the scope of the tempo induction algorithms themselves.

Secondly, the evaluation data sets used by many researchers are usually private and of relatively small size, which makes it difficult to compare one system with another. For example, a collection of score-matched MIDI performance data is available from the Music, Mind and Machine Group of

F. Gouyon is with the Universitat Pompeu Fabra, `fgouyon@iua.upf.es`
A. Klapuri is with the Tampere University of Technology, `klap@cs.tut.fi`
S. Dixon is with the Austrian Research Institute for Artificial Intelligence, `simon@oefai.at`
M. Alonso is with the École Nationale Supérieure des Télécommunications, `miguel.alonso@enst.fr`
G. Tzanetakis is with Victoria University, `gtzan@cs.uvic.ca`
C. Uhle is with the Fraunhofer Institute for Digital Media Technology, `uhle@idmt.fraunhofer.de`
P. Cano is with the Universitat Pompeu Fabra, `pcano@iua.upf.es`

the University of Nijmegen[1] (around 200 performances of a couple of Beatles songs by 12 pianists performed in several tempo conditions). Results on this data set were reported by Cemgil et al. [5] and Dixon [6], and the latter argued that more challenging data was needed. Also, Temperley [3] provides a publicly available data set[2] of 46 pieces with metronomical timing and 16 performed pieces, all taken from the common-practice Western repertoire. However, in both cases, the data sets are only suitable for evaluating systems dealing with MIDI input, and not acoustic signal input.

As a first step towards more systematic evaluations and comparisons, a contest was organised during the International Conference on Music Information Retrieval (ISMIR 2004) held at the University Pompeu Fabra in Barcelona in October 2004.[3] The task was restricted to the induction of tempo as a scalar, in beats per minute (BPM), and not the individual beat positions. Researchers were encouraged to participate by several means, including a "call for algorithms" on the Music Information Retrieval mailing-list;[4] the respondents set up and agreed upon a common evaluation benchmark for the competition.

In this paper, we present 11 of the 12 algorithms tested and highlight differences in their implementations —note that even if most of these algorithms are often referred in the current literature as state-of-the-art algorithms, we acknowledge that they do not represent a comprehensive coverage of work in beat induction (see [1] for a review). We then detail the evaluation framework set up for the contest, the test database and the evaluation method. The results are then presented and discussed, with a focus on relating the performance differences to design choices in the systems. We stress important achievements in the field of audio tempo induction, and highlight open issues and possible avenues for future work in this field, proposing ways to tackle them in further, improved, tempo induction contests.

For information on other audio description contests, also held during ISMIR 2004 (on Genre Classification, Artist Identification, Melody Detection and Rhythm Classification), see [7] and the contest webpage.

II. Algorithms

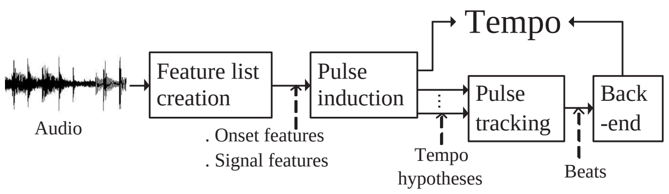


Fig. 1. General tempo induction computational scheme.

Of the 12 algorithms entered in the contest, 11 were submitted by 6 different research teams, and one open-source algorithm (GPL-licensed) was downloaded from the web. One entrant chose not to participate in this report, so we report here on 11 algorithms, which are described below in alphabetical order. The contest organiser did not compete.

Algorithms were submitted in various formats: the open-source entries were submitted as C, C++ or Matlab source code, and the others as Windows or GNU/Linux binaries or Matlab pre-parsed pseudocode files.

All of the algorithms are based on a common general scheme: a *feature list creation* block, that parses the audio data into a temporal series of features which convey the predominant rhythmic information to the following *pulse induction* block (see Figure 1 and [1] for more details). The features can be onset features or signal features computed at a reduced sampling rate. For example, onset features might consist of a list of times and amplitudes of note onsets, whereas signal features might consist of average energy values computed on successive 10 ms or 20 ms frames, or a differential of the energy in various frequency bands.

Many algorithms also implement a *beat tracking* block. However, as the contest did not address the issues of tracking tempo changes and determining beat positions, the submitted algorithms either bypassed this block or added a subsequent back-end for the purpose of the contest, i.e. a parsing of the beat positions into a global tempo estimation.

A. Alonso

Miguel Alonso from the École Nationale Supérieure des Télécommunications (ENST) in Paris submitted two algorithms, referred to as `AlonsoACF` and `AlonsoSP`, which were submitted in the form of p-files, i.e. Matlab pre-parsed pseudocode files (source code is not visible).

Both methods are based on the same front-end that extracts phenomenal accents, i.e., onsets of notes, by detecting sudden changes in dynamics, timbre, or harmonic structure. A time-frequency representation of the audio signal is calculated, and the rate of change of the spectral energy content is found by filtering this representation with a differentiator FIR filter. The positive contributions of each spectral line are summed in the frequency domain and a quasi-periodic and noisy pulse-train is obtained. This pulse-like signal exhibits sharp maxima at those time instants where a phenomenal accent occurs.

The difference between the systems is found in the pulse induction block. The first method is based on the autocorrelation of the pulse signal, while the latter uses the spectral product. Both algorithms are described in detail in [8]. These systems were originally conceived for beat-tracking, but the tracking part was disabled in the versions submitted to the contest.

B. Dixon

Simon Dixon from the Austrian Research Institute for Artificial Intelligence (ÖFAI) in Vienna submitted three entries to the contest: `DixonI`, `DixonT` and `DixonACF`.

The first two are GNU/Linux binaries based on the beat tracking system BeatRoot detailed in [9].[5] They are both based

[1]`http://www.nici.kun.nl/mmm/archives/`

[2]`ftp://ftp.cs.cmu.edu/usr/ftp/usr/sleator/melisma2003`

[3]The conference webpage is `http://ismir2004.ismir.net/`

[4]`http://listes.ircam.fr/wws/info/music-ir`

[5]BeatRoot is available as GPL code at `http://www.oefai.at/~simon/beatroot/index.html`

on a simple energy-based onset detector followed by an inter-onset interval (IOI) clustering algorithm. DixonI selects a tempo based on the "best" cluster, where the clusters are assessed by the number of IOIs they contain, the amplitude of the corresponding notes, and the support of other clusters related by simple integer ratios. DixonT selects several prominent clusters as tempo hypotheses, performs beat tracking based on these hypotheses, and outputs the mean of the inter-beat intervals (IBI) from the best beat tracking solution as the final estimate of tempo.

DixonACF (Matlab source code) is described in [10]. This algorithm splits the signal into 8 frequency bands, and then smooths, downsamples and performs autocorrelation on each of the frequency bands. From each band, the 3 highest peaks (excluding the zero-lag peak) of the autocorrelation function are combined, and each is assessed as a possible tempo candidate, with the highest scoring peak determining the final tempo value.

C. Klapuri

Anssi Klapuri from the Tampere University of Technology submitted one algorithm as a GNU/Linux binary, referred to as Klapuri.

An important aspect of this algorithm lies in the feature list creation block: the differentials of the loudness in 36 frequency subbands are combined into 4 "accent bands", measuring the "degree of musical accentuation as a function of time." The goal in this procedure is to account for subtle energy changes that might occur in narrow frequency subbands (e.g. harmonic or melodic changes) as well as wide-band energy changes (e.g. drum occurrences). The pulse induction block implements a bank of comb filters comparable to that proposed by Scheirer [11] (see below).

Another particularity of this algorithm is the joint determination of three metrical levels (the tatum,[6] the beat and the measure) through probabilistic modelling of their relationships and temporal evolutions. After computing the beats of the whole test excerpt, the tempo was computed as the median of the IBIs of the excerpt's latter half. See [13] for a complete description of the algorithm.

D. Scheirer

The source code of Eric Scheirer's algorithm (formerly MIT Media Lab) was downloaded from the web (http://sound.media.mit.edu/~eds/beat/tapping.tar.gz). Anssi Klapuri ported it to GNU/Linux —it is the same code that was used in Klapuri's evaluations [13]— and it was then compiled in the UPF labs (it is referred to as Scheirer).

An important novelty promoted in [11] was to perform pulse induction on regularly-sampled series of signal features (amplitude envelope) rather than on series of discrete events (as onset times). Further, Scheirer also argued that pulse induction should be performed *separately* on the signal features computed on each of several frequency bands, and then combined, rather than on a single series containing the combined features. In the implementation used, the number of frequency subbands is 6.

Another important novelty was to introduce the use of comb filterbanks for the pulse induction block. This technique is foreshadowed by the "clock model" of Povel and Essens [14] in that it seeks the series of periodically-spaced clock pulses that best matches the feature list (the implementation in the form of a bank of comb filters introduces an exponential decay on the clock pulse amplitudes).

The output of the algorithm is a set of beat times rather than an overall tempo estimate, so we added a small back-end to the code that outputs the state of the filterbank after the analysis of the whole sound file. Then the tempo is taken to be the resonance frequency of the filter with the highest instantaneous energy after the whole analysis. The choice of this particular back-end was based on the observation that this algorithm provides more reliable estimates after some processing of the sound file than at the beginning. However, note that other methods could also be considered, as for instance, the total number of beats divided by the duration, or the mean of the IBIs. See [11] for more details on the algorithm.

E. Tzanetakis

George Tzanetakis from Victoria University submitted 3 entries: TzanetakisH, TzanetakisMS and TzanetakisMM (standing respectively for "Histogram", "MedianSum" and "MedianMultiband"). GNU/Linux binaries were compiled at the UPF labs from the source code available on the SourceForge web.[7]

All the three methods are based on the wavelet front-end described in [15]. The signal is segmented in time into 3 s analysis windows (with an overlap of 1.5 s). In each window, the signal is decomposed with the help of a wavelet transform into 5 octave-spaced frequency bands, and the amplitude envelope is extracted in each band.

Regarding the pulse induction block, all three methods use autocorrelation, however, they differ in some aspects. The default method (TzanetakisMS) sums the diverse subband amplitude envelopes and computes an autocorrelation of the resulting sum. The maximum peak in the autocorrelation (a tempo estimate) is computed on each analysis window and the median of the tempo estimates is chosen as the final tempo. TzanetakisMM makes a separate tempo estimate for each band and each analysis window, and then selects the median. TzanetakisH sums the subband amplitude envelopes, computes an autocorrelation of the resulting sum, selects several autocorrelation peaks and accumulates them in a histogram which summarises the peaks of all analysis windows. The tempo is finally set to the highest peak of the histogram.

F. Uhle

Christian Uhle from Fraunhofer Institute for Digital Media Technology submitted one algorithm as a Windows binary, referred to as Uhle.

[6]fastest metrical level, i.e. the regular time division that most highly coincides with all note onsets [12, p.22]

[7]marsyas-0.2 under http://www.sourceforge.net/projects/marsyas

This algorithm calculates the rates of metrical pulses on three levels (the tatum, the beat and the measure). The audio signal is segmented into characteristic long-term segments corresponding for example to a verse or a chorus [16]. Amplitude envelopes for logarithmically spaced frequency bands are calculated by means of the Discrete Fourier Transform and smoothed using an FIR low-pass filter. Slope signals of the amplitude envelopes are computed by means of the relative difference function, as suggested in [13], and half-wave rectification. The slope signals are summed across all bands to produce an "accent signal." An autocorrelation function (ACF) is computed for non-overlapping 2.5 s segments inside each long-term segment. The tatum period is estimated from the ACF by means of a periodicity detection procedure; and a second ACF is calculated on a larger time scale (7.5 s) to detect periodicities in the range of musical measures. A function representing periodicity saliences at integer multiples of the tatum period (i.e. ACF local maxima) is computed and compared (i.e. correlated) with a number of pre-defined metrical templates, which characterise musical knowledge of different meters. The current implementation has 17 templates. The most highly correlated template determines the value of the segment's tempo. Tempi are accumulated in a weighted histogram and the maximum yields the basic tempo of the piece. See [17] for more details.

III. Experimental framework

A. Infrastructure

Two computers were used: `AlonsoACF`, `AlonsoSP` and `Uhle` were run on Windows OS (XP Professional edition 2002, version 5.1.2600), the rest on GNU/Linux OS (Debian Sarge), both 1.6 GHz, with 512 MB RAM. The evaluation framework was designed as a set of Matlab (version 6.1, Release 12.1 on GNU/Linux and version 6.5, Release 13 on Windows), perl, shell and dos scripts. For a robustness test (see below), several types of distortion were applied to the signal using the programs Sox and Matlab. However, it was ensured that the tempo was still clearly perceivable even in the cases of severe degradation of signal quality. All of the test scripts are available from the contest webpage.

B. Data

No training data was provided. However, some preparatory data (7 instances and corresponding tempo values) was given to the participants in order to compare whether algorithms yield the same output when run in participants' labs and on UPF machines, and to check proper formatting of algorithm input and output.[8]

The test data consisted of 3199 tempo-annotated instances in 3 data sets as described below. The instances range from 2 to 30 seconds, and from 24 BPM[9] to 242 BPM. Figure 2 illustrates the distribution of test excerpts along the tempo axis (these tempo statistics are all available in text format on the contest webpage and may be used e.g. for setting prior probabilities in Bayesian approaches). They all have approximately constant tempi, and the format is the same for all: mono, linear PCM, 44100 Hz sampling frequency, 16 bit resolution. The total duration of the test set is approximately 45140 s (i.e. around 12 h 36 min). This data was not available to participants before the competition. Part of the data has now been made available on the contest webpage.

1) Loops: Many sound libraries are made up of short "loops" to be used in DJ sessions, or for home recording needs. The loops used here were originally in MP3 format, they come from different sound library retailers and are courtesy of the Tape Gallery.[10] It is usual that tempo in BPM (and additional metadata) are sold together with sound files. These annotations were not double-checked. We do not distribute these loops for copyright reasons, however, an exhaustive list of loops and corresponding tempo is available on the contest webpage.[11] One can search by name for, listen to MP3 versions of and buy high audio quality versions of any of these loops from the webpage of the Tape Gallery.

A loop is often used as a basic short "kernel" to be looped in a composition, that is, to generate a long audio file by several concatenations of the same instance. However, the samples used in the analysis were not looped.

- Total number of instances: 2036
- Duration: a few bars
- Total duration: around 15170 s
- Tempo range: between 60 and 215 BPM, see Figure 2(c)
- Genres: Electronic, Rock, House, Ambient, Techno.

2) Ballroom: BallroomDancers.com[12] provides information on ballroom dancing (online lessons, etc.). Some characteristic excerpts of many dance styles are provided in real audio format, labelled with a tempo value. Tempo values were double-checked by the third author.

Data and annotations are available on the contest webpage.

- Total number of instances: 698
- Duration: around 30 s
- Total duration: around 20940 s
- Genres: see style distribution in Table I.
- Tempo range: between 60 and 224 BPM, see Figure 2(b)

Style	**# instances**
Cha Cha	111
Jive	60
Quickstep	82
Rumba	98
Samba	86
Tango	86
Viennese Waltz	65
Slow Waltz	110

TABLE I
STYLE DISTRIBUTION OF THE BALLROOM DANCE MUSIC EXCERPTS

[8]This was not considered as "training data" as it would not be possible to properly train a system with so few instances and test it on a test set more than 400 times greater.

[9]Note however that only 15 excerpts have a tempo less than 50 BPM

[10]`http://www.sound-effects-library.com/`

[11]`http://www.iua.upf.es/mtg/ismir2004/contest/tempoContest/node4.html`

[12]`http://www.ballroomdancers.com/`

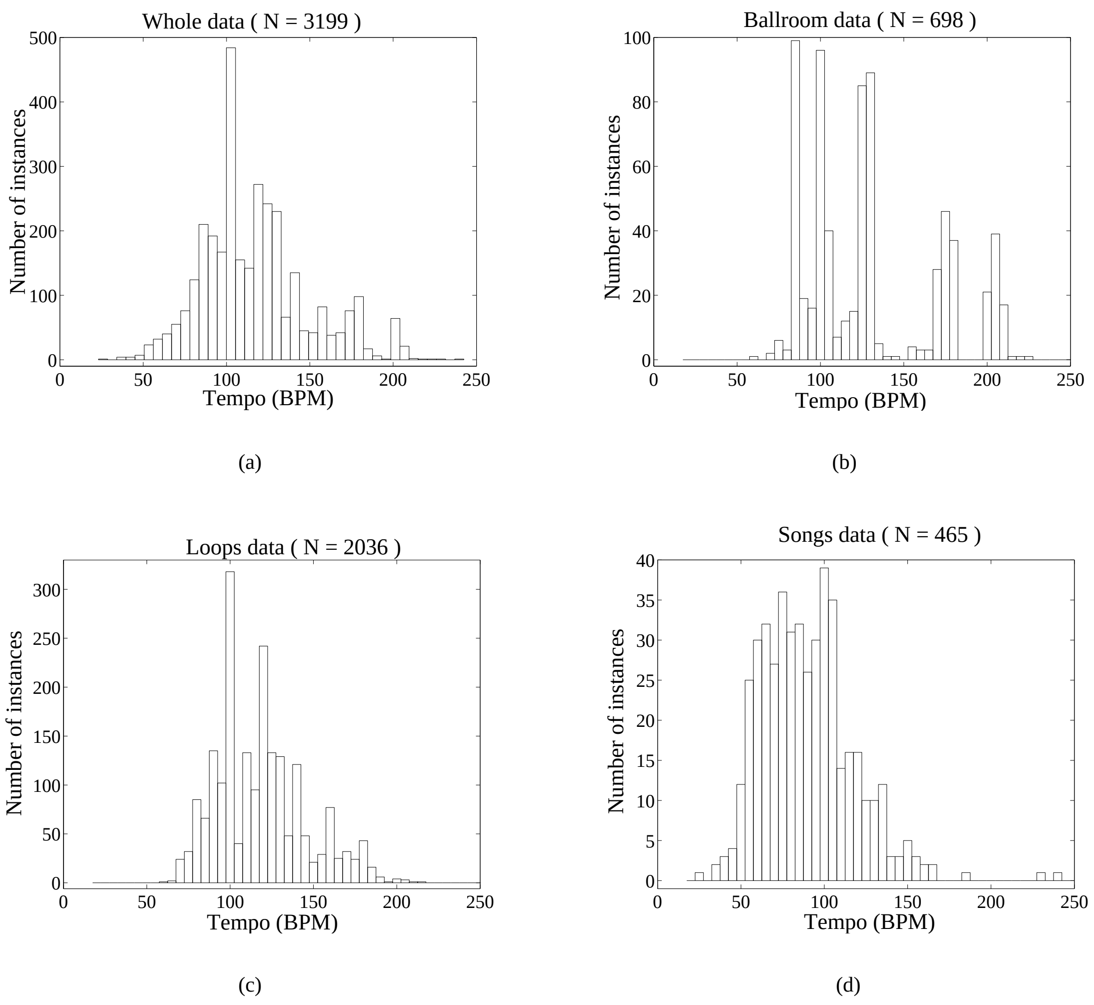


Fig. 2. Histograms of ground-truth tempo values in 5 BPM steps

3) Song excerpts: A professional musician placed beat marks on several song excerpts. (These beats were cross-checked by the first author). The ground-truth tempo was computed as the median of the IBIs; here also, other methods could also be considered, as for instance, the total number of beats divided by the duration.

Data and annotations are available on the contest webpage.

- Total number of instances: 465
- Duration: around 20 s
- Total duration: around 9300 s
- Genres: see distribution in Table II.
- Tempo range: between 24 and 242 BPM, see Figure 2(d)

Genre	**# instances**
Rock	68
Classical	70
Electronica	59
Latin	44
Samba	42
Jazz	12
AfroBeat	3
Flamenco	13
Balkan and Greek	144

TABLE II
GENRE DISTRIBUTION OF THE SONG EXCERPTS

C. Evaluation methods

Two evaluation metrics were agreed for the contest:

- *Accuracy 1*: The percentage of tempo estimates within 4% (the *precision window*) of the ground-truth tempo.
- *Accuracy 2*: The percentage of tempo estimates within 4% of either the ground-truth tempo, or half, double, three times, or one third of the ground-truth tempo.

The latter evaluation metric was motivated by the fact that the ground-truth we use for evaluation does not necessarily represent the metrical level that the majority of human listeners would choose. However, we assume that discrepancies between ground-truth tempo and human perception correspond to a focus on a different metrical level, i.e., a ratio of 2 or $\frac{1}{2}$ for duple meter music and a ratio of 3 or $\frac{1}{3}$ for triple meter music. This assumption is ubiquitous in all previous evaluation attempts; see V-A for further discussion. As we discuss in further details in IV-B.1 and V-A, the width of the precision window is not a crucial factor.

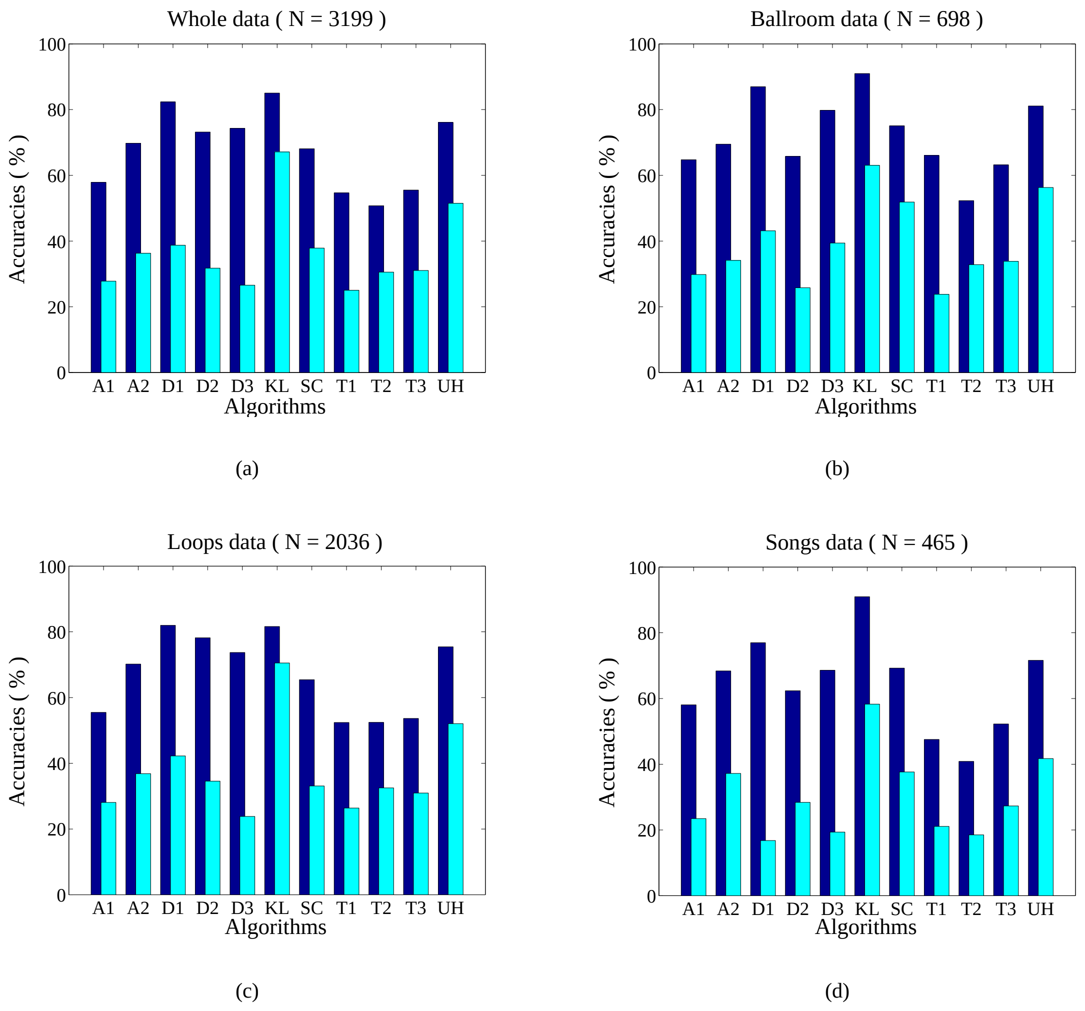


Fig. 3. Accuracies 1 (light) and 2 (dark) on the whole data set –3(a)–, the Ballroom data set –3(b)–, the Loops data set –3(c)– and the Songs data set –3(d).

In addition, the robustness of algorithms to sound distortion was evaluated on a part of the test data: the 465 song excerpts. Test instances were distorted by several processes: downsampling/resampling, GSM encoding/decoding, filtering, volume change and addition of reverberation and white noise (with a signal-to-noise ratio of 20 dB). The script is available on the contest webpage and in the appendix.

IV. Results

A. *Accuracy measures and robustness to noise*

Figure 3 presents the results for each algorithm, ordered alphabetically: A1 is `AlonsoACF`, A2 is `AlonsoSP`, D1 is `DixonACF`, D2 is `DixonI`, D3 is `DixonT`, KL is `Klapuri`, SC is `Scheirer`, T1 is `TzanetakisH`, T2 is `TzanetakisMM`, T3 is `TzanetakisMS` and UH is `Uhle`. For each algorithm, accuracy 1 and 2 are given, in light and dark shadings, respectively, for the whole data set and each of the 3 subsets.

Figure 4 illustrates the loss of accuracy for each algorithm when distortion was applied to the Songs data set as detailed above. Clearly, algorithms `AlonsoACF`, `AlonsoSP`, `DixonI` and `DixonT` suffer more from distortions than other algorithms.

1) And the winner is...: The performance measures accuracy 1 and 2 were the criteria used to determine the contest winner. As can be seen in Figure 3, the algorithm `Klapuri` outperformed the others with respect to these measures on all data sets: respectively 67.29% and 85.01% on the whole data set and {70.71%, 81.57%}, {63.18%, 90.97%} and {58.49%, 91.18%} on the Loops, Ballroom and Songs data sets, respectively. It was also the best algorithm with respect to noise robustness (loss of 1.72 percentage points in accuracy 2, see Figure 4).

2) Statistical significance: One must keep in mind that, because of the restriction to a specific data set, the numbers reported in Figure 3 are just *estimates* of the true (but unknown) algorithm accuracies. Therefore, in addition to providing success rates for each algorithm, it is important to consider whether the observed differences in performance are statistically significant or arise by chance.

Different statistical tests can be used to compare algorithms based on their respective predictive accuracy: e.g. a test for the difference of two proportions, Student's *t* test, McNemar's test,

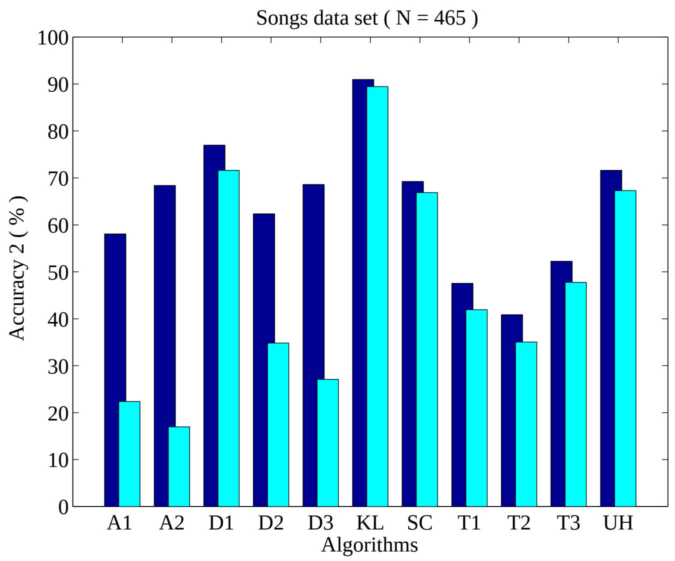


Fig. 4. Effect of instance distortions on accuracy 2, dark bars for clean data, light bars for distorted data.

cross-validation paired differences *t* tests [18]. Choosing the appropriate test to a given problem depends on the suitability of several assumptions, among them independence of algorithm accuracies (i.e. accuracies on test items are independent for algorithm A and algorithm B) and error independence between items (i.e. errors made by an algorithm on two separate test items are independent).[13]

In our problem, algorithms are all tested on the same instances, therefore we cannot assume that, for a given instance, the failures of different algorithms are independent. On the other hand, it seems reasonable to assume that errors made by a specific algorithm on different instances are independent. As mentioned in [19, Paragraph 3.2] and [18, Question 3], McNemar's test is appropriate to this kind of problem.

McNemar's statistical test tests the hypothesis that the fact that algorithm A classifies an item correctly while algorithm B classifies it incorrectly is equally likely as the opposite (algorithm B classifies an item correctly while algorithm A classifies it incorrectly). In other words it tests the fact that given only one algorithm makes an error, it is equally likely to be either one (this is the "null hypothesis"). Given a threshold for statistical significance (usually 0.01 or 0.05) the null hypothesis is tested by applying a two-tailed test with a Normal distribution (see [19] for more details).

According to this statistical test, the observed difference (of around 1%) in accuracy 1 on the whole data set (see Figure 3(a)) between `AlonsoACF` and `DixonT` would arise by chance on 19% of occasions, this difference is therefore not statistically significant (considering a p-value of 0.01 as the threshold for statistical significance), and it is better to conclude that both algorithm performances are comparable. Similarly, observed performance differences between `AlonsoSP` and `DixonACF` (less than 3%), `AlonsoSP` and `Scheirer` (less than 2%), `DixonACF` and `Scheirer` (1%), `DixonI` and `TzanetakisMM` (1%), `DixonI` and `TzanetakisMS` (less than 2%), `DixonT` and `TzanetakisH` (less than 2%) and `TzanetakisMM` and `TzanetakisMS` (less than 1%) are not statistically significant, setting the threshold for significance to a p-value of 0.01.[14] The differences between all remaining pairs of algorithms are representative of genuine performance differences.

Regarding accuracy 2, solely the differences between `AlonsoACF` and `TzanetakisMS` (less than 3%), `AlonsoSP` and `Scheirer` (less than 2%), `DixonI` and `DixonT` (less than 1%), `DixonT` and `Uhle` (less than 2%) and `TzanetakisH` and `TzanetakisMS` (less than 1%) are not significant.[15] The differences between all remaining pairs of algorithms are statistically significant.

3) Computation time: Another interesting aspect of the algorithms is the computational resources they require. It can be expressed as processing time divided by excerpt length:[16] `DixonI` takes approximately 0.02 times the excerpt length to estimate its tempo, `DixonT`, `Uhle`, `AlonsoSP` and `AlonsoACF` approximately 0.1, `Scheirer` approximately 0.4, `Klapuri` approximately 0.5, `DixonACF` approximately 1 and `TzanetakisH`, `TzanetakisMM` and `TzanetakisMS` approximately 2. (Note that the participants were not instructed to optimise computational efficiency when submitting the algorithms and using different operating systems and versions of Matlab may have an influence on computation time.)

To facilitate comparison of other algorithms with those of the contest, detailed results on each dataset are available on the contest webpage.[17] In the following sections, we provide a more detailed analysis of the results, done after the publication of the contest results in October 2004.

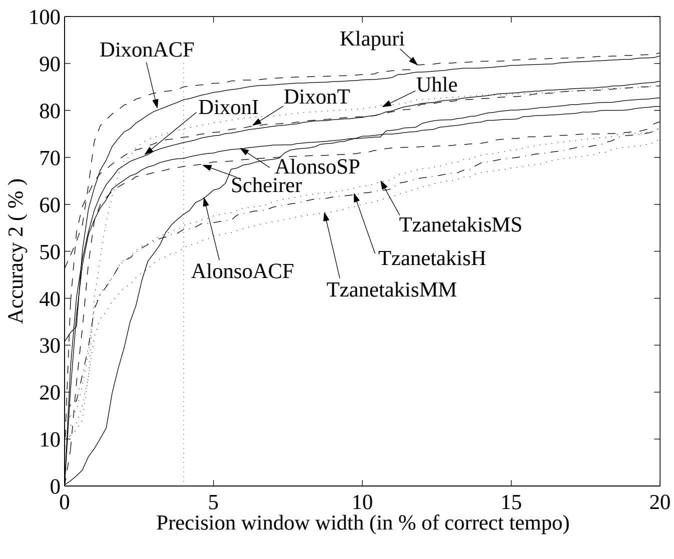


Fig. 5. Accuracy 2 vs precision window width, full data set

[13]which does not mean that algorithm accuracy would be independent of the test set

[14]They correspond respectively to P-values of 0.02, 0.16, 0.46, 0.25, 0.5, 0.13 and 0.5.

[15]P-values of 0.03, 0.09, 0.18, 0.03 and 0.26

[16]Algorithm computation times are approximately proportional to excerpt length.

[17]`http://www.iua.upf.es/mtg/ismir2004/contest/tempoContest/Results.htm`

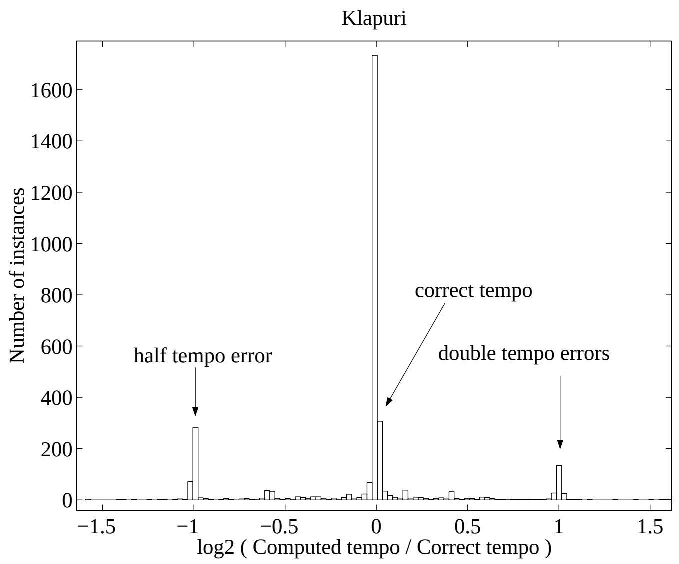


Fig. 6. Histogram of ratio between estimated tempo and correct tempo for `Klapuri`, full test data set

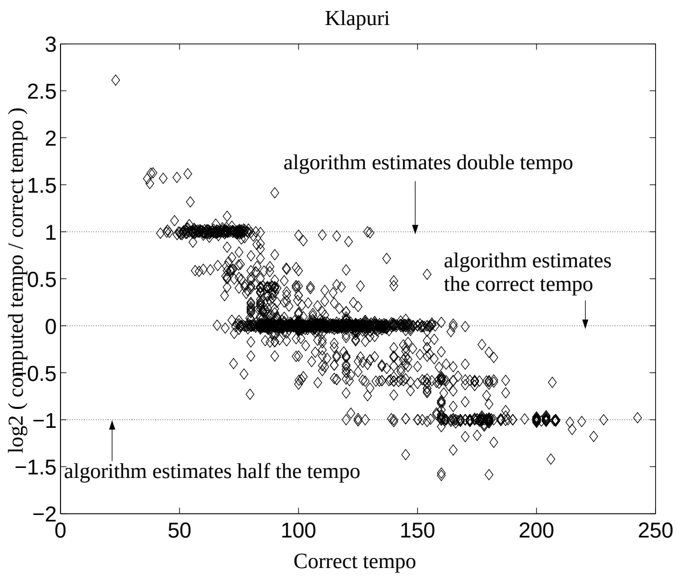


Fig. 7. `Klapuri` performance with respect to instance tempi, full test data set

B. Error analysis

1) Accuracy vs precision window width: Figure 5 plots the relationship of algorithm performance to precision window width. The choice of 4% precision in accuracies 1 and 2 is somewhat arbitrary. In the literature, other values have been advocated; for instance, Klapuri et al. [13] propose a precision of 17.5% for IBIs, however they focus on *consecutive* IBIs rather than on global tempo and deal with excerpts with varying tempo. The amount of tempo variation in the data is an important factor to consider in setting the precision. Since we are dealing with basically constant-tempo data, a small precision window seems appropriate.

2) Tendencies towards integer ratio errors: Figure 6 shows the type of errors made by the contest winner (`Klapuri`). Figure 7 shows the same information plotted against tempo. One can see on the one hand that the most common errors are doubling and halving of tempo, and on the other hand that it shows a "moderate tempo tendency", i.e. a tendency to estimate half the tempo for fast pieces and double for slow pieces. We remark also that it estimates incorrectly (with respect to accuracy 1) all pieces whose tempi lie outside the rough limits of 60 to 160 BPM. This is due to the explicit modelling of a prior probability function for the tempo [13], [20].

Regarding the other algorithms, inspection of their error histograms also shows clearly that, as expected, halving and doubling of tempo are the most common errors. On the other hand, `Klapuri` seems to be the only algorithm that clearly shows a moderate tempo tendency. With the exception of `Klapuri` and `Scheirer`, all algorithms tend to "tap too fast" rather than too slow. For instance, as can be seen in Figures 8(a) and 8(b), `DixonT` has a very clear tendency towards faster metrical levels.

Other typical error factors are $\frac{4}{3}$ and $\frac{2}{3}$, as seen, for example, in the peaks around -0.58 and 0.41 on the (logarithmically-scaled) X-axis of Figures 8(a), 8(b) and 8(c). An error of $\frac{4}{3}$ in the tempo estimation represents an error of $\frac{3}{4}$ in the IBI, that is, a focus on e.g. the dotted quarter-note instead of the half-note, while a tempo error of $\frac{2}{3}$ represents a focus on e.g. the dotted-quarter note instead of the quarter-note.

Algorithms also sometimes estimate $\frac{1}{3}$ of the correct tempo. See, for instance, the peak around -1.58 in Figures 8(c) and 8(a). This error factor, as well as 3 and $\frac{3}{2}$, are typical of triple and compound meter pieces (e.g. Waltz in the Ballroom data set). We found relatively few of these errors, presumably because relatively few such pieces are present in the test data set.

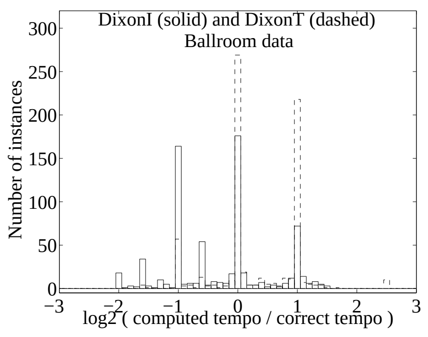
3) Algorithm performance "niches": It is interesting to consider whether specific algorithms, regardless of their overall performance, show unique performance on some particular data. Indeed, an algorithm which performs worse than other algorithms on many problems, but solves a few problems that no other algorithm solves, would be valuable if these special cases could be identified.

There are 41 pieces (3 ballroom, 35 loops and 3 songs) whose tempi were correctly computed by *all* 11 algorithms. On the other hand, 176 pieces (11 ballroom, 162 loops and 3 songs) were incorrectly processed by *all* algorithms, with respect to both accuracy 1 and accuracy 2. Finally, there are 29 pieces whose tempi were correctly computed by a *single* algorithm. No clear explanations for these cases have been found.

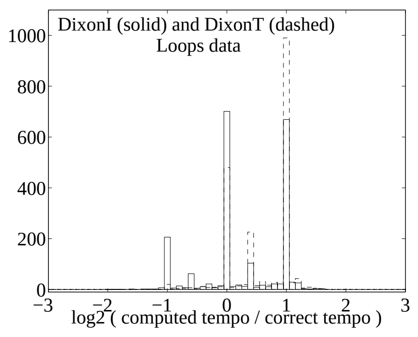
Another way to thoroughly inspect the results is to compare pairs of algorithms. For instance, Figure 9 shows a comparison between `Klapuri` and `DixonACF`. For each data set, instances have been ordered with respect to increasing error made by `Klapuri`, where the error is computed as follows:

$$abs(log2(computedTempo/correctTempo)).$$

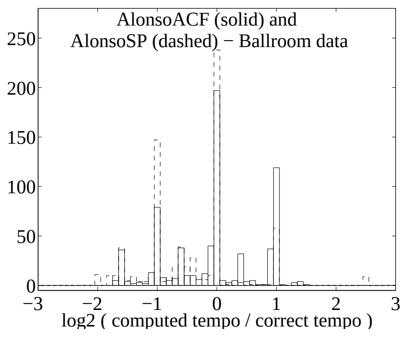
The performance of both algorithms is given for each instance, permitting a visual comparison of algorithms on an instance by instance basis. Three main trends are apparent: many cases



(a)



(b)



(c)

Fig. 8. Histograms of ratio between estimated tempo and correct tempo for `DixonI` (solid) and `DixonT` (dashed), on the Ballroom and Loops data sets —8(a) and 8(b) respectively— and for `AlonsoACF` (solid) and `AlonsoSP` (dashed), on the Ballroom data set —8(c).

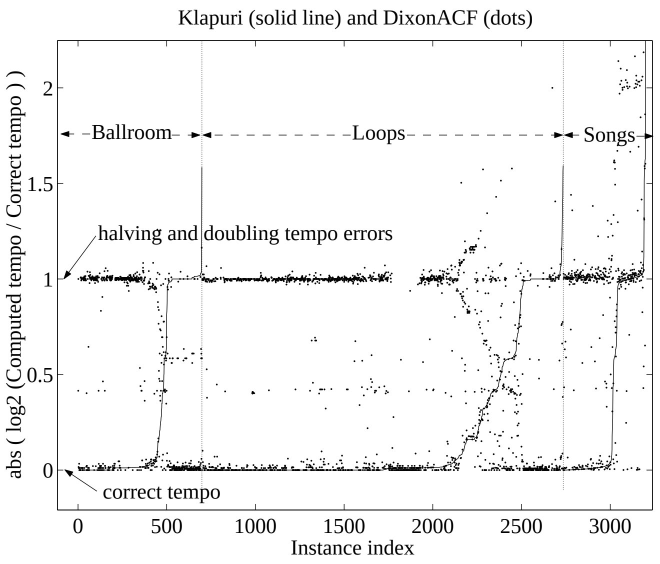


Fig. 9. Comparison of errors made by `Klapuri` (solid line) and `DixonACF` (dots)

of agreement between the algorithms, for correct and incorrect tempo estimates; cases where one algorithm is correct and the other has a halving or doubling error; and cases where both algorithms are incorrect, and one algorithm has double the tempo of the other.

For example, in the Ballroom data, `DixonACF` solves quite a few doubling and halving errors that `Klapuri` makes (see the cluster of points around the error value of 0 for indexes between 500 and 698), but on this very data it also makes quite a few doubling or halving errors where `Klapuri` estimates the correct tempo. This is also true of the Loops data set (indexes between around 2500 to 2734), but not the Songs data set, where `DixonACF` makes many doubling and halving errors (this can also be seen in Figure 3(d), third bar pair from the left). On the other hand, `DixonACF` seems to solve some non-integer ratio errors that `Klapuri` makes, especially in the Loops data set (indexes between around 2100 and around 2500, where `Klapuri`'s error on the Y-axis is between 0 and 1). Note that the apparent mirroring of error values (reflection in the line $y = 0.5$) is an artifact of the representation, which occurs when one algorithm has a log error e, where $-1 < e < 0$, and the other algorithm has double this tempo, hence a log error of $e' = e + 1 = 1 - |e|$.

Figures such as Figure 9 can be generated for any pair of algorithms. They show on an instance by instance basis which errors an algorithm makes that another one does not make. We can then track down single files for which a specific algorithm has a particular advantage over another one. Cases where several algorithms make the same error could be used to identify interesting ("pathological") test cases for further investigation, general weaknesses in current tempo induction systems, and errors in annotation. However, in order to draw conclusions about error trends, or alternatively, specific "skills" or "performance niches" of algorithms, much more test data is needed, together with richer metadata. Indeed, it is difficult to make any valid conclusion just by listening to or examining specific test cases.

C. *Redundant approach to tempo estimation*

Having several algorithms performing the same task and exhibiting specific performance on specific parts of the data, an important question arises: Can we improve the tempo estimation accuracy by combining the outputs of several algorithms? The answer seems to be "yes", although it should be noted that simply computing e.g. the median of the tempo estimates of different algorithms does *not* yield an improvement. This is because the "too slow" and "too fast" tempo estimates cannot be guaranteed to balance each other out.

A thorough analysis of algorithm skills and error trends would dictate a set of rules for combining algorithms. Lacking this information, we propose in the following a voting mechanism for combining the tempo estimates of different algorithms. Imagine an ordered list of a number of algorithms among the 11. Algorithms in the list being considered in turn, for each test instance, a given algorithm gets one vote from all the algorithms that agree with its tempo estimate. An algorithm X is defined to agree with an algorithm Y if the ratio of their estimates is 1, 0.5, or 2, with 4% precision. The tempo estimate of the algorithm which gets the largest number of votes among the algorithms is selected as the output. If several algorithms receive the same number of votes, the order of the algorithms in the list defines which estimate is selected as the output.

An exhaustive search over all possible combinations of five algorithms (from among the 11) was made to find a combination which performs best using the voting mechanism. Applying the accuracy measure 1, the algorithms [`Klapuri`, `Uhle`, `Klapuri`, `DixonI`, `DixonACF`] achieved 68% performance and, applying accuracy measure 2, the algorithms [`Klapuri`, `Scheirer`, `DixonT`, `DixonI`, `DixonACF`] achieved 86% performance. This does not represent a significant improvement compared to the performance of `Klapuri` alone (67% and 84% according to the accuracy measures 1 and 2, respectively). However, the situation becomes clearer when `Klapuri` is excluded. In this case, the algorithms [`Uhle`, `Scheirer`, `DixonI`, `DixonACF`, `DixonT`] together achieve a rate of 57% with accuracy 1 and the algorithms [`Scheirer`, `Uhle`, `DixonT`, `DixonACF`, `DixonACF`] achieve a rate of 84% with accuracy 2. Compared to the best individual performances among the remaining algorithms (`Uhle` achieves 51% with accuracy 1 and `DixonACF` achieves 81% with accuracy 2), the voting mechanism makes a statistically significant improvement to the individual results.

The experiment described above is an example of a "redundant" approach to music content analysis: instead of designing one very complex algorithm we combine a number of different and more simple mechanisms. This idea stems from Bregman, who pointed out that human perception appears to be redundant at many levels: there are several different processing principles serving the same purpose, and when one of them fails, another succeeds [21].

V. Discussion

Let us now discuss further these results and the evaluation benchmark itself.

A. On accuracy measures

Accuracy 2 was designed to account for the inherent fuzziness of the tempo induction task: two listeners might not agree on a metrical level as the "correct" tempo. However, its drawback (in our use of it) is that it does not take the meter into account. Considering half and double ground-truth tempo as correct makes sense solely for duple meter instances. Similarly, considering three times and one third of ground-truth tempo as correct makes sense solely for instances with a triple or compound meter. The meter is not available with the data used here. We therefore considered half, double, three times, and one third of ground-truth tempo as correct. However, the test data does not contain many triple or compound meter pieces, so the inclusion of the factors 3 and $\frac{1}{3}$ in the computation of accuracy measure 2 was perhaps not justified.

One reason to choose a wide precision window is the approximate nature of ground-truth annotations. Figure 5 does not indicate any significant difference between the 4% precision used here and wider windows. However, 4% is probably the highest precision level that should be considered as the Just-Noticeable Difference (JND) for tempo differences is approximately 4% for music [22]. Further, larger precision windows may be required to evaluate *consecutive* IBIs rather than global tempo, especially in varying tempo situations [13].

B. Onsets vs frame features

In Figure 4, we can see that algorithms `AlonsoACF`, `AlonsoSP`, `DixonI` and `DixonT` clearly suffer more from distortion of the signal than other algorithms. These 4 algorithms are the only ones that attempt to detect onsets of discrete sound events as a first step. All others measure some physical feature in the input signal in a more continuous (frame-based) manner.

Therefore, implementing a robust tempo induction algorithm calls for the computation of low-level frame features rather than that of onset lists as the first processing block. However, whether this is perceptually more valid (as proposed in [11]) remains to be investigated.

C. Towards better benchmarks

1) Beat tracking: Tempo induction and beat tracking are part of the same perceptual process [23], therefore future evaluation efforts should consider them jointly.

2) Data: More data is needed for future contests. Importantly, a larger amount of data with triple and other meters is required. However, not all music is suitable. As discussed and exemplified in [6], test instances can show diverse levels of difficulty. It may be difficult to induce or track the tempo of specific musical pieces, even from constant-tempo performances, if they have a complex rhythmic structure (e.g. many events not on beats, or many beats occurring between musical events), while other pieces may be fundamentally less challenging. Additionally, in the case of performed music, keeping an almost steady tempo or adding expressive tempo variations is up to the performer. For instance, results here are better on Ballroom data; this was predictable as this is dance music, which has relatively clear beats and stable tempo.

Therefore, measuring the level of "rhythmic difficulty" of the instances in the test set might provide an additional control for thorough evaluations. Goto and Muraoka [4] and Dixon [6] propose such metrics.

3) Robustness tests: Other robustness tests are needed, for instance, robustness to increasing levels of noise (decreasing SNR) and robustness to cropping (the effect of the length of the excerpt).

4) Better annotations and evaluation measures: It is difficult to evaluate the accuracy of an algorithm for determining the correct tempo because of the inherent ambiguity of metrical levels. In future contests, more accurate evaluations might be obtained by considering the "degree of ambiguity" of excerpt tempi. This could be done by recruiting several annotators (e.g. 3 or 4) for each piece and considering several metrical levels as valid options *only* in cases where annotators disagree on the tempo. This procedure could also tell us *which* levels are valid for each instance. This is however a very time-consuming procedure.

A faster way to proceed to more precise evaluations would be to manually annotate beats of each test instance at *2 different metrical levels* instead of one. A single person would suffice for annotating each instance. For each instance, the accuracy measure of an algorithm would be the best match over the annotated metrical levels.

One might object that, for a given instance, two algorithms might not be evaluated with respect to the same metrical level. Nevertheless, both levels have been considered valid by the annotator. And we can assume that, in tempo-ambiguous cases, any two listeners would perceive at least one level in common, solely the rankings of metrical levels would differ. Consider the following example: a piece of music whose levels all share duple relationships, to which listener A taps the beat at 50 BPM. Being asked to define another level, he chooses 100 BPM (it is highly unlikely that he would choose 25 BPM which is too slow to be a perceptually valid tempo). Say that listener B naturally taps the beat at 200 BPM, being asked to define another level, he will most likely choose 100 BPM (not 400 BPM). Even in this extreme case, there exists some agreement. Thus, this procedure would be a way to measure *how close a specific algorithm gets to human agreement regarding tempo perception*. Such annotations could be done with the help of annotation tools as proposed e.g. in [9] and [24].

5) More modular evaluations: It is difficult to compare systems that, even if they implement similar concepts, do not share any piece of code. The performance of each system depends on the overall implementation and it is often hard to say anything more than "system A performed better than system B (on this data set)." That is, we are unable to say anything conclusive about the system submodules (for instance, whether frame differentials are better than absolute values), without being able to switch the submodules within a single system. On the other hand, it would be difficult to implement different systems in a common software framework so that they share simple processing blocks. Indeed, forcing the use of a specific implementation framework would probably have negative repercussions in terms of the number of contest entries. In the evaluations detailed above, different system variants from the same participant (Alonso, Dixon or Tzanetakis) give the most reliable information about the effect on the performance of different solutions for a given submodule of the system. A solution could therefore be to motivate participants to submit several systems, with small, but conceptually relevant, variations in some submodules.

D. Open issues

In future contests, with more data, better annotation, more elaborate robustness tests and evaluation measures and most importantly more modular evaluations, it would be possible to evaluate the following open issues more thoroughly:

1) Periodicity detection before or after multiband integration?: Current literature, e.g. [11], [13], advocates the use of multiband processing and subsequent integration of periodicity estimates, rather than periodicity estimation after the integration of a signal processed in several frequency bands. For example, Scheirer argues that "a rhythmic processing algorithm should treat frequency bands separately, combining results at the end, rather than attempting to perform beat-tracking on the sum of filterbank outputs" [11].

The difference between `TzanetakisMS` and `TzanetakisMM` lies precisely in the integration of several frequency bands respectively before or after periodicity estimation. The algorithms exhibit similar performance when assessed with accuracy 1, but the former performs around 5% better than the latter with respect to accuracy 2. It is difficult to make any solid conclusion and confirm or refute Scheirer's point from these results. Let us however outline a few aspects of these methods: Estimating periodicities after multiband integration enhances periodicities that are present in *all* bands, while periodicity estimation before multiband integration favours signals whose periodicities appear solely in a restricted frequency region. Also, the former method has a bias towards fast metrical levels; indeed, it accounts for the phase of periodicities while the latter does not. Consider for example the case where two bands have the same periodicity but have a phase difference of half the period: the former method yields double the tempo of the latter. This is verified on the data used here: `TzanetakisMS` makes more double-tempo errors than `TzanetakisMM`.[18] One can argue that each method is more suitable for different types of data. Further evaluations are required before more general conclusions can be drawn.

2) Which frequency decomposition?: Scheirer argues that his algorithm "is not particularly sensitive to the particular bands" [11]. That is, the important point is to proceed to a frequency decomposition, and not the particular choice of decomposition.

However, let us consider the algorithms that compute periodicities in frequency subbands (`DixonACF`, `Klapuri`, `Scheirer` and `TzanetakisMM`). They all use energy (or integrated amplitude) features. Of course, the performance of each system depends on the overall system, so it is hard to say anything conclusive about the best frequency decomposition (as indeed about any of the submodules). However, the fact that these systems show non-negligible differences in performance suggests that the definition of the frequency filterbank could be a significant issue, contrary to Scheirer's observation.

Further, many features other than energy could be computed on signal frames. Energy could also be normalised (or not) in each frequency band. This also suggests that more research is needed on the definition of relevant rhythmic features.

3) Frame values vs differential values: Some pulse induction algorithms focus on energy values (e.g. `TzanetakisMM`) while others focus on changes from one frame to the next (e.g. estimating the derivative of frame energy values, e.g. `Klapuri`, or of the downsampled amplitude envelope, e.g. `Scheirer`). The derivative can be estimated by a first-order differentiator filter (as for `Scheirer`) or more accurately as proposed in [8]. If, as Klapuri claims in [13], we assume that the difference between the use of the autocorrelation and that of comb filterbanks for pulse induction is not crucial in the performance of a tempo induction system, the performance of `Scheirer` vs that of `TzanetakisMM`[19] seems to indicate that changes in energy values would be more valuable

[18]Note that accuracy 2 does not considered them as errors

[19]Respectively 37% vs 30% with accuracy 1 and 68% vs 50% with accuracy 2

rhythmic features than the energy values themselves. However, here also, a solid conclusion would require implementations to differ solely in this aspect.

4) Which pulse induction method?: There are significant differences in the accuracies obtained by `AlonsoSP` and `AlonsoACF`, which differ solely in the pulse induction block. The spectral product outperforms the autocorrelation on all data sets and all accuracy measures.[20] This finding should be verified on other data sets as Alonso's results seem to indicate different conclusions (namely that the autocorrelation would be better than spectral product [8, Tables 2 and 3, p.162]). A comparison with a comb filterbank method (used by the contest winner) would also be interesting.

5) Induction vs tracking: It is sometimes hypothesised that in order to compute a tempo value that best reflects human perception of the musical pace, it would be better to consider the whole tracking process rather than rely solely on tempo induction [25]. Performance differences between `DixonT` and `DixonI` are not really conclusive in that respect. On this point also, more research is needed.

A short comment should also be made regarding the back-end added to `Scheirer`'s output. The final tempo was taken to be the resonance frequency of the filter with the highest instantaneous energy at the end of the whole sound file analysis. One might wonder whether this is being unfair to this algorithm. On the one hand, if the analysis fails at the very end of the sound file, the overall tempo might be wrong while most beats were correctly tracked. On the other hand, the rather slow exponential decay used by this algorithm tends to yield more reliable estimates at the end of the file than at the beginning (at least with constant-tempo data, as used here).

6) Joint estimation of several metrical levels: Three algorithms (`Klapuri`, `Uhle` and `DixonACF`) implement, in different ways, influential schemes for the determination of 2 or 3 metrical levels. As they all perform very well, it seems interesting to evaluate more methodically the effect of this feature.

Similarly, the relevance of the "moderate tempo tendency" that has to be considered when focusing simultaneously on several levels, and often modelled with a prior tempo probability function (as in [20]), should also be the object of further research.

7) Redundant approach: From the results presented in IV-C, it appears that combinations of algorithms can perform better than any single algorithm. This is an interesting avenue for future work and raises the following interesting questions: Which commonalities and differences should we implement in the concurring algorithms? How simple should we keep these algorithms? Is the voting scheme proposed in this paper the best way to combine algorithms?

An interesting way to tackle this problem could be to embrace a machine learning perspective and focus on ensemble learning methods. In supervised learning, ensemble learning algorithms take decisions regarding the membership of a given instance to a class among several possible classes by considering the "votes" of several classifiers, previously trained on labelled data. Designing several classifiers for the same task can be done in several manners, for instance it is possible to provide different subsets of the training data to a base algorithm (as is done e.g. in *Bagging* and *Boosting*), it is also possible to provide the same training data but described with different attributes [26].

To use these methods for tempo induction, one would need to define training and test data sets and, possibly, discretise tempo in a number of classes. Then, forcing diversity in the design of different algorithms could be done by specialising each of them on a restricted set of signal features. Another option could be to focus the performances of different algorithms on different training instances (or differently-weighted instances, which would amount to *Boosting*).

VI. Summary

Quantitative comparisons of tempo induction algorithms are largely absent from the literature. The contest reported in this document was aimed at promoting more systematic evaluations. It is our hope that the contest results, data and annotations might be useful in developing future benchmarks.

This evaluation showed that, for music with almost constant tempo, tempo induction is feasible with around 80% accuracy and a relatively good robustness to distortion, if we do not insist on finding a specific metrical level. Anssi Klapuri's algorithm is the best among the algorithms tested. We encourage other researchers to participate in future benchmarks.

It also showed that the most common errors that all algorithms make are in the choice of metrical level. The majority of algorithms tend to tap too fast rather than too slow. Tests of robustness to signal distortions showed that robust tempo induction entails the processing of frame features rather than that of onset lists.

However, emulating the perception of tempo by humans is still an unsolved problem. Inducing the basic tempo from arbitrary audio signals, without accepting alternative metrical levels, is not a solved issue, and many aspects call for further research. Here is a (non-exhaustive) list of open issues: Should periodicity detection be performed before or after multiband integration? Which frequency decomposition is most appropriate? Which rhythmic features are the most relevant to compute from audio as a first processing step? Is it better to use absolute frame values or differential values? Which pulse induction method performs best? Is it better to consider the whole tracking process rather than relying solely on tempo induction, in order to better emulate human perception of the musical pace? Should several metrical levels be estimated jointly? What is the effect of implementing a moderate tempo tendency? How can we combine several algorithms effectively?

With this article, we wish to stimulate future benchmarks. For these, we argue that beat tracking should be evaluated jointly with induction and that better annotations, more robustness tests, better evaluation metrics and more modular evaluations are needed.

[20]Note that it is however more sensitive to distortion

APPENDIX
SOUND DISTORTION SCRIPTS

System commands:

- Resampling:
```
$ sox wavfile.wav -r8000
soxedfile0.wav rate
```
- GSM encoding/decoding and upsampling:
```
$ sox soxedfile0.wav soxedfile1.gsm
$ sox soxedfile1.gsm -sw -r44100
soxedfile2.wav rate
```
- Filtering and volume adjustment:
```
$ sox soxedfile2.wav soxedfile3.wav
filter 500-2000
$ sox soxedfile3.wav soxedfile4.wav
vol 1.8
```
- Reverb application:
```
$ sox soxedfile4.wav soxedfile5.wav
reverb 1 2000 1000 700 750 760 880
```

Matlab commands:

- White noise addition:
```
>> [x,fs,bits] = wavread(
soxedfile5.wav);
>> SNR = 20;
>> Px = sum(sum(x.^2));
>> noise = rand(size(x))-.5;
>> Pnoise = sum(sum(noise.^2));
>> noisyX = x + noise*sqrt(
(Px/Pnoise) * 10^(-SNR/10) );
>> wavwrite(noisyX, fs, bits,
tempwavfile.wav);
```

ACKNOWLEDGEMENTS

The authors would like to thank three anonymous reviewers for useful and detailed suggestions. The Loops data set was kindly provided to the UPF by the Tape Gallery as research material during the AudioClas[21] project. The first author also wishes to thank Giorgos Emmanouil for annotating the Songs data set, other UPF people involved in other audio description contests (Markus Koppenberger, Nicolas Wack, Emilia Gómez, Bee Suan Ong and Sebastian Streich) as well as Kris West for the Matlab code of the McNemar test and Martin McKinney for comments. This research was supported by the EU project FP6-507142 SIMAC.[22] ÖFAI thanks the BMBWK and BMVIT for financial support.

REFERENCES

[1] F. Gouyon and S. Dixon, “A review of automatic rhythm description systems,” *Computer Music Journal*, vol. 29, no. 1, pp. 34–54, 2005.

[2] F. Gouyon and B. Meudic, “Towards rhythmic content processing of musical signals: Fostering complementary approaches,” *Journal of New Music Research*, vol. 32, no. 1, pp. 41–64, 2003.

[3] D. Temperley, “An evaluation system for metrical models,” *Computer Music Journal*, vol. 28, no. 3, pp. 28–44, 2004.

[4] M. Goto and Y. Muraoka, “Issues in evaluating beat tracking systems,” in *Proc. IJCAI Workshop on Music and AI*. International Joint Conference on Artificial Intelligence, 1997, pp. 9–16.

[21]`http://audioclas.iua.upf.edu`
[22]`http://www.semanticaudio.org`

[5] T. Cemgil, B. Kappen, P. Desain, and H. Honing, “On tempo tracking: Tempogram representation and kalman filtering,” *Journal of New Music Research*, vol. 29, no. 4, pp. 259–273, 2001.

[6] S. Dixon, “An empirical comparison of tempo trackers,” in *Proc. 8th Brazilian Symposium on Computer Music*. Brazilian Computing Society, 2001, pp. 832–840.

[7] P. Cano, E. Gómez, F. Gouyon, P. Herrera, M. Koppenberger, B. Ong, X. Serra, S. Streich, and N. Wack, “ISMIR 2004 audio description contest,” *To appear*.

[8] M. Alonso, B. David, and G. Richard, “Tempo and beat estimation of musical signals,” in *Proc. International Conference on Music Information Retrieval*. Barcelona: Audiovisual Institute, Pompeu Fabra University, 2004, pp. 158–163.

[9] S. Dixon, “Automatic extraction of tempo and beat from expressive performances,” *Journal of New Music Research*, vol. 30, no. 1, pp. 39–58, 2001.

[10] S. Dixon, E. Pampalk, and G. Widmer, “Classification of dance music by periodicity patterns,” in *Proc. International Conference on Music Information Retrieval*, 2003, pp. 159–165.

[11] E. Scheirer, “Tempo and beat analysis of acoustic musical signals,” *J. Acoust. Soc. Am.*, vol. 103, no. 1, pp. 588–601, 1998.

[12] J. Bilmes, “Timing is of the essence: Perceptual and computational techniques for representing, learning, and reproducing expressive timing in percussive rhythm,” Masters Thesis, MIT, Cambridge, 1993.

[13] A. Klapuri, A. Eronen, and J. Astola, “Analysis of the meter of acoustic musical signals,” *IEEE Trans. Speech and Audio Processing*, 2005.

[14] D. Povel and P. Essens, “Perception of temporal patterns,” *Music Perception*, vol. 2, no. 4, pp. 411–440, 1985.

[15] G. Tzanetakis and P. Cook, “Musical genre classification of audio signals,” *IEEE Trans. Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[16] J. Foote, “Automatic audio segmentation using a measure of audio novelty,” in *Proc. IEEE Conference on Multimedia and Expo*, 2000, pp. 452–455.

[17] C. Uhle, J. Rohden, M. Cremer, and J. Herre, “Low complexity musical meter estimation from polyphonic music,” in *Proc. AES 25th International Conference*. New York: Audio Engineering Society, 2004, pp. 63–68.

[18] T. Dietterich, “Approximate statistical tests for comparing supervised classification learning algorithms,” *Neural Computation*, vol. 10, no. 7, pp. 1895–1924, 1998.

[19] L. Gillick and S. Cox, “Some statistical issues in the comparison of speech recognition algorithms,” in *Proc. IEEE Conference on Acoustics, Speech and Signal Processing*, 1989, pp. 532–535.

[20] R. Parncutt, “A perceptual model of pulse salience and metrical accent in musical rhythms,” *Music Perception*, vol. 11, no. 4, pp. 409–464, 1994.

[21] A. Bregman, “Psychological data and computational ASA,” in *Computational Auditory Scene Analysis*, D. Rosenthal and H. Okuno, Eds. New Jersey: Lawrence Erlbaum Associates, 1998.

[22] A. Friberg and J. Sundberg, “Time discrimination in a monotonic, isochronous sequence,” *Journal of the Acoustical Society of America*, vol. 98, no. 5, pp. 2524–2531, 1995.

[23] P. Desain and H. Honing, “Computational models of beat induction: The rule-based approach,” *Journal of New Music Research*, vol. 28, no. 1, pp. 29–42, 1999.

[24] F. Gouyon, N. Wack, and S. Dixon, “An open-source tool for semi-automatic rhythmic annotation,” in *Proc. International Conference on Digital Audio Effects*, 2004, pp. 193–196.

[25] F. Gouyon, S. Dixon, E. Pampalk, and G. Widmer, “Evaluating rhythmic descriptors for musical genre classification,” in *Proc. AES 25th International Conference*. New York: Audio Engineering Society, 2004, pp. 196–204.

[26] T. Dietterich, “Ensemble learning,” in *The Handbook of Brain Theory and Neural Networks*, 2nd ed., M. Arbib, Ed. Cambridge MA: MIT Press, 2002.