A POST-PROCESSING PROCEDURE FOR IMPROVING MUSIC TEMPO ESTIMATES USING SUPERVISED LEARNING

Hendrik Schreiber
tagtraum industries incorporated
hs@tagtraum.com

Meinard Müller
International Audio Laboratories Erlangen
meinard.mueller@audiolabs-erlangen.de

ABSTRACT

Tempo estimation is a fundamental problem in music information retrieval and has been researched extensively. One problem still unsolved is the tendency of tempo estimation algorithms to produce results that are wrong by a small number of known factors (so-called octave errors). We propose a method that uses supervised learning to predict such tempo estimation errors. In a post-processing step, these predictions can then be used to correct an algorithm's tempo estimates. While being simple and relying only on a small number of features, our proposed method significantly increases accuracy for state-of-the-art tempo estimation methods.

1. INTRODUCTION

Tempo-related tasks are well established in *music information retrieval* (MIR) [1]. One common task is to estimate the *tempo* humans "tap" along to a beat when listening to music. Another task, *beat tracking*, attempts to determine the exact times at which beats occur. In this paper, we deal with tempo estimation exclusively. While in some genres—like Romantic music—local tempo changes are common [11], Pop, Rock, and Dance music often have one steady, global tempo, i.e. it can be represented by a single number usually specified in *beats per minute* (BPM). The method proposed in this paper is only suitable for music with such a global tempo.

Over the years, many different approaches to tempo estimation have been taken. Gouyon et al. [9] provided a comparative evaluation of the systems that participated in the ISMIR 2004 contest. Five years later, Zapata and Gómez gave an updated overview [30]. To our knowledge, the most recent comprehensive evaluations are presented in [2, 23, 24]. For a textbook-style introductory overview describing different approaches, see [20] by Müller.

Many methods divide the estimation problem into two phases. First, via an *onset strength signal* (OSS) or novelty curve, beat candidates are found. Second, one or more periodicities are extracted from the OSS. Methods for finding periodicities are often based on the Fourier transform, but also include autocorrelation [23], tempograms [29], the interonset interval (IOI) histograms [26], and resonating comb filters [2]. The decision for a final result is based on simple heuristics, genre classification [15, 25], secondary tempo estimates [24], the discrete cosine transform of IOI histograms [6], or a feature-based learning approach like *Gaussian mixture models* (GMM) [22], *support vector machines* (SVM) [8, 23], *k-nearest neighbor classification* (k-NNC) [29], and neural networks [5].

For evaluation, results are typically compared with a ground truth allowing a 4% tolerance. This measure is called *Accuracy1*. Because many algorithms have a tendency to under- or over-estimate the true value by a factor of 2 or 3, a second measure called *Accuracy2* has been introduced. *Accuracy2* allows for errors that correspond to a factor of 2, 3, ½, or ⅓, also known as *octave errors*. Despite evidence that algorithms as well as humans can distinguish between slow and fast music [13, 18], *Accuracy1* values for state-of-the-art algorithms are still below *Accuracy2*. One way to change this may lie in genre- or style-related knowledge [4]. Many genres are partially defined by a certain tempo or tempo range, which can be exploited to pick the right octave. Schuller et al. [25] demonstrated this for the Ballroom dataset and Hörschläger et al. [15] did the same for the GiantSteps tempo dataset. The fact that the excellent method by Böck et al. [2] scores remarkable 95% when trained on the Ballroom dataset with 8-fold cross validation, but reaches only 66.8% on the mixed genre GTZAN dataset further supports this notion.

In this paper we describe a supervised learning approach to correct common tempo estimation errors. This is achieved by re-framing error correction as a classification problem. We are able to demonstrate that the proposed method performs better or as well as state-of-the-art algorithms when combined with a simple tempo estimation method. Furthermore, because our error correction approach can be trained for any tempo detection method, we are able to show improvements in *Accuracy1* for previously published algorithms via post-processing.

The remainder of this paper is structured as follows: in Section 2 we describe a simple tempo estimation algorithm, test datasets, measures, and investigate common estimation error classes. Then, in Section 3, we explain our post-processing procedure, which corrects tempo estimates using supervised learning based on a small number of audio features. In Section 4 we evaluate the proposed features, and then compare our results with those from other methods. Finally, in Section 5, we present our conclusions.

2. TEMPO ESTIMATION

To lay the groundwork for our error correction method, we first describe a simple tempo estimation algorithm, then introduce several test datasets and discuss common pitfalls. In Section 2.5, we introduce performance metrics and describe observed errors.

2.1 Algorithm

To estimate the dominant pulse we follow the approach taken in [24], which is similar to [23, 28]: We first convert the signal to mono and downsample to 11025 Hz. Then we compute the power spectrum Y of 93 ms windows with half overlap, by applying a Hamming window and performing an STFT. The power for each bin $k \in [0:K] := \{0, 1, 2, \ldots, K\}$ at time $m \in [0:M] := \{0, 1, 2, \ldots, M\}$ is given by $Y(m,k)$, its positive logarithmic power $Y_{\ln}(m,k) := \ln(1000 \cdot Y(m,k) + 1)$, and its frequency by $F(k)$ given in Hz. We define the onset signal strength $\mathrm{OSS}(m)$ as the sum of the bandwise differences between the logarithmic powers $Y_{\ln}(m,k)$ and $Y_{\ln}(m-1,k)$ for those k where the frequency $F(k) \in [30, 720]$ and $Y(m,k)$ is greater than $\alpha Y(m-1,k)$ (see [16]):

$$\begin{aligned} I(m,k) &= \begin{cases} 1 & \text{if } Y(m,k) > \alpha Y(m-1,k) \\ & \text{and } F(k) \in [30, 720], \\ 0 & \text{otherwise} \end{cases} \qquad (1) \\ \mathrm{OSS}(m) &= \sum_k \left(Y_{\ln}(m,k) - Y_{\ln}(m-1,k)\right) \cdot I(m,k) \end{aligned}$$

Both the factor $\alpha = 1.76$ and the frequency range were found experimentally [24].

The $\mathrm{OSS}(m)$ is transformed using a DFT with length 8192. At the given sample rate, this ensures a resolution of 0.156 BPM. The peaks of the resulting beat spectrum B represent the strength of BPM values in the signal [7], but do not take harmonics into account [10, 21]. Therefore we derive an enhanced beat spectrum B_{E} that boosts frequencies supported by harmonics:

$$B_{\mathrm{E}}(k) = \sum_{i=0}^{2} \left| B(\lfloor k/2^i + 0.5 \rfloor) \right| \qquad (2)$$

Similar to an enhanced beat histogram [28], B_{E} incorporates harmonics by simply adding to each bin the magnitudes of the bins denoted by half and by a quarter of its own frequency—or, if not available—the closest available bin. We choose to use fractions instead of multiples for modeling harmonics and thus essentially model the fourth harmonic, not the first. This allows us to take advantage of the full DFT resolution without oversampling, as each bin for the first harmonic is mapped to four different bins for the fourth harmonic. To estimate the tempo T of the dominant pulse (or periodicity in the OSS), we determine the

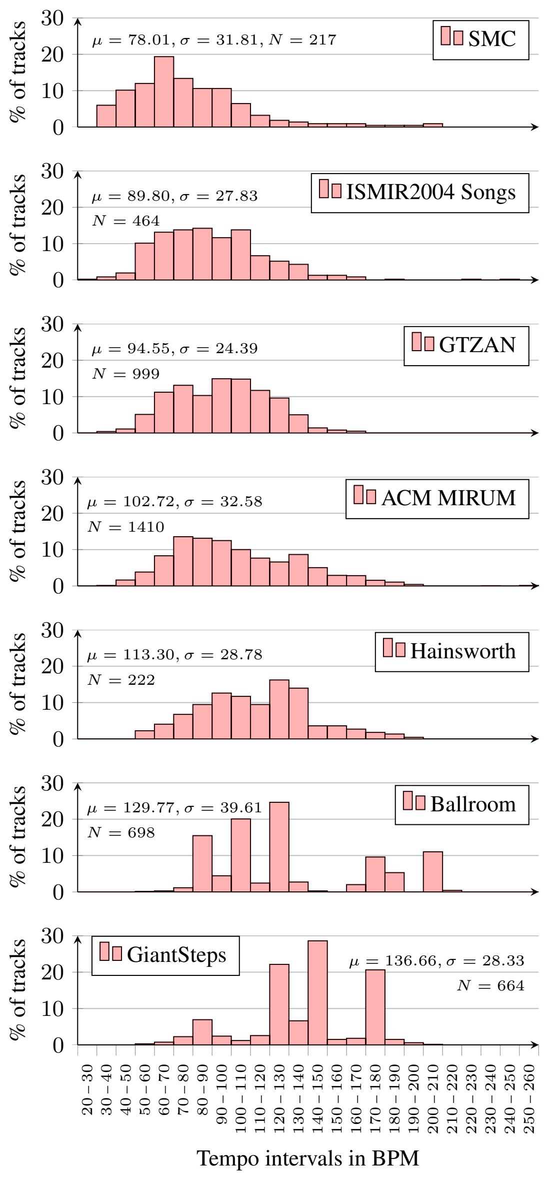

Figure 1. Tempo distributions for the test datasets.

highest value of B_{E}, divide its frequency by 4 to find the first harmonic, and finally convert its associated frequency to BPM:

$$T = F(\operatorname*{argmax}_k B_{\mathrm{E}}(k)) \cdot \frac{60}{4} \qquad (3)$$

To ensure meaningful results for most kinds of Western music, we constrain T to $[40, 250]$ by halving or doubling its value, if necessary.

2.2 Test Datasets

It has become customary to benchmark tempo estimation methods with results reported for a small set of well known datasets. These are ACM MIRUM [18, 22], Ballroom [9], GTZAN [27], Hainsworth [12], ISMIR04 Songs [9], and SMC [14]. The latter was specifically designed to be diffi-

Dataset	E_0	E_1	E_2	$E_{1/2}$	E_3	$E_{1/3}$	E_4	$E_{1/4}$	$E_{3/2}$	$E_{2/3}$	$E_{5/4}$	$E_{4/5}$	$E_{4/3}$	$E_{3/4}$
ACM˙MIRUM	0.7	73.5	16.0	7.0	0.8	0.0	0.1	0.0	1.8	0.1	0.0	0.0	0.0	0.1
Ballroom	0.4	64.3	1.0	29.7	0.0	1.1	0.0	0.7	0.1	2.3	0.0	0.0	0.0	0.3
Hainsworth	8.6	64.4	1.4	19.4	0.0	0.5	0.0	0.0	1.8	1.8	0.9	0.5	0.5	0.5
GTZAN	4.0	72.2	15.7	5.1	0.4	0.1	0.0	0.0	1.0	0.4	0.1	0.4	0.5	0.1
ISMIR04 Songs	4.3	64.7	19.4	6.3	1.1	0.2	0.0	0.0	2.4	0.4	0.4	0.2	0.2	0.4
SMC	29.0	37.8	6.0	10.6	0.0	2.3	0.0	0.0	5.1	2.3	0.5	2.3	0.9	3.2
Combined	3.9	68.1	12.3	11.2	0.5	0.4	0.0	0.1	1.5	0.8	0.1	0.3	0.2	0.4
GiantSteps	7.1	63.1	4.1	21.5	0.0	0.0	0.0	0.0	0.6	0.3	0.8	0.9	0.5	1.2

Table 1. Error class distribution for tempo estimates T (given in BPM) for different datasets in percent.

Dataset	Sweet Oct.	Cov.	90%	95%
ACM MIRUM	$69-138$	72.8	$50-152$	$50-170$
Ballroom	$71-142$	71.1	$84-204$	$82-204$
Hainsworth	$79-158$	82.4	$58-150$	$57-167$
GTZAN	$66-132$	80.9	$55-130$	$52-138$
ISMIR04 Songs	$59-118$	74.1	$48-131$	$36-136$
SMC	$51-102$	68.7	$32-115$	$32-143$
Combined	$69-138$	72.9	$40-150$	$50-180$
GiantSteps	$91-182$	88.1	$85-175$	$80-180$

Table 2. Sweet octaves and their respective coverage in percent for the test datasets (left). Shortest BPM intervals required to achieve a test set coverage of 90% or 95% (right).

cult for beat trackers. Where applicable, we used the corrected annotations from [23]. We refer to the union of these six datasets as the *Combined* dataset. Additionally, we test against the recently published GiantSteps dataset for electronic dance music (EDM) [17]. It is not included in *Combined* to allow direct comparisons with older literature.

Not surprisingly, all mentioned datasets differ in their composition (Figure 1). The mean tempo ranges from $78\ \mathrm{BPM}$ (SMC) to $137\ \mathrm{BPM}$ (GiantSteps) and the standard deviation spans from 24 (GTZAN) to 40 (Ballroom). Furthermore, the tempo distributions of Ballroom and GiantSteps contain some distinct spikes, while the other datasets more closely resemble normal distributions. None of the datasets have uniformly distributed tempi.

2.3 Octave Bias

If a dataset's tempo distribution is not uniform and most values fall into a relatively small interval, constraining results to this interval may lead to fewer octave errors. We call deliberately choosing such an interval *octave bias*.

To illustrate this, assume an algorithm for the GiantSteps dataset with 50% *Accuracy1*, but 100% *Accuracy2*. Further assume that all errors are by a factor of 2 or $1/2$. 88.1% of all tempi in GiantSteps happen to be in $[91, 182)$. If we constrained results to this interval by halving and doubling, *Accuracy1* would increase from 50% to 88.1%.

Each described dataset has such a *sweet octave*, i.e. a tempo interval $[j, 2j)$ that contains more of the dataset's songs than any other octave (Table 2). In the absence of a uniform test set, it is therefore important to test the same algorithm against datasets with different sweet octaves, thus revealing the effects of octave bias. On the positive side, a specialized or genre-aware algorithm may benefit from exploiting knowledge about the test dataset (e.g. EDM-specific tempi [15]). Additionally to sweet octaves, Table 2 lists the shortest BPM intervals required to achieve a certain test set coverage. For example, to cover 90% of the tempi in the ACM MIRUM test set, one only needs to look at the interval $[50, 152]$ and not at the considerably larger interval $[37, 257]$ required for full coverage.

2.4 Genre Bias

While octave bias describes how algorithms can exploit constraining results to certain tempo intervals, *genre bias* describes a technique for algorithms to constrain their output to a relatively small set of distinct tempi that are characteristic for the genres in the dataset.

A good example for this is the Ballroom dataset. Even though the dataset contains 698 songs, only 63 different tempi occur. Assuming an unbiased algorithm with integer precision is constrained to the $[40, 250]\ \mathrm{BPM}$ interval, it solves a task equivalent to choosing one out of 210 classes. An algorithm trained on the Ballroom dataset using k-fold cross validation "knows" that there are only 63 classes and therefore has a considerably easier task to solve.

2.5 Measures

As mentioned above, tempo estimation algorithms are usually evaluated with two metrics: *Accuracy1*, defined as the percentage of correct estimates with 4% tolerance, and *Accuracy2*, the percentage of correct estimates ignoring errors caused by the factors 2, 3, $1/2$, and $1/3$.

Because we aim to correct estimation errors, we need to test our tempo estimation method against the test datasets and record not just accuracies, but also the kinds of errors. To do so, we define the error classes E_2, E_3, E_4, $E_{3/2}$, $E_{5/4}$, $E_{4/3}$ and their reciprocals with the index indicating the error factor. Just like *Accuracy1* and *Accuracy2* we allow a 4% tolerance. Since not all estimates are wrong and some errors are not covered by the mentioned classes, we define E_1 for correct estimates (equivalent to *Accuracy1*) and E_0 for errors not described otherwise. This leads to a total of 14 classes forming the label set $\mathbb{E} := \{E_0, E_1, \ldots\}$. Table 1 shows the distribution of estimated tempi over $\mathbb{E}$ for the test datasets using the tempo estimation method from Section 2.1. For *Combined*, 12.3% of all tempi are in E_2, while 11.2% are in $E_{1/2}$. Only 3.9% of all estimated values cannot be explained by one of the defined factors and thus are collected under the label E_0. This implies an upper bound of 96.1% *Accuracy1* for any error correction scheme based on $\mathbb{E}$ w.r.t. the *Combined* datasets.

3. TEMPO ERROR CORRECTION

As we have seen, most wrong tempo estimates are off by a limited number of factors. Therefore the correction of T can be re-framed as a classification problem, which is solvable using supervised machine learning. Knowing the error class for an estimate then allows us to calculate the true tempo. In the following subsections we describe the features used for classification, the training dataset, and the tempo correction procedure.

3.1 Features

In order to keep the algorithm simple, we use as features only T and a very small set of audio features. While not attempting to specifically model genres, the features we use aim at characterizing rhythm, tonality and beat intensity. Combined, we expect them to capture essential information about a musical piece.

3.1.1 Log Beat Spectrum

The tempi corresponding to the most common estimation classes $E_{1/2}$, E_1, and E_2 fall onto a logarithmic scale. To mirror this, we use a *logarithmic beat spectrum* (LBS) to describe the different periodicities in the signal. LBS is computed by resampling/interpolating B into 10 logarithmically spaced bands representing tempi ranging from 40 to 500 BPM. Subsequently it is normalized so that its highest value is 1.

While LBS provides a more complete picture of the periodicities than just the dominant tempo T, it does not add any information about the frequency bands these periodicities stem from. As a second modification to B, we create different versions of LBS based on the five slightly overlapping bands $[30, 110]$, $[100, 220]$, $[200, 440]$, $[400, 880]$ and $[800, 1600]$ Hz. Combined, these spectra form the *multiband logarithmic beat spectrum* (LBS_M). For a given song, LBS_M consists of $5 \times 10 = 50$ features.

3.1.2 Spectral Flatness

To represent tonality we use *spectral flatness* (SF), also known as Wiener entropy. It is defined as the ratio between the geometric and the arithmetic mean of the power spectrum:

$$\text{SF}(m) = \frac{\left(\prod_{k=0}^{K-1} Y(m,k)\right)^{\frac{1}{K}}}{\frac{1}{K}\sum_{k=0}^{K-1} Y(m,k)} \qquad (4)$$

To determine $\text{SF}(m)$, we re-use the power spectra $Y(m, k)$ already computed in Section 2.1. For increased robustness against low sample rates, we limit k to $F(k) \in [30, 3000]$. As the two features for a given song we use both the mean and the variance of all its $\text{SF}(m)$.

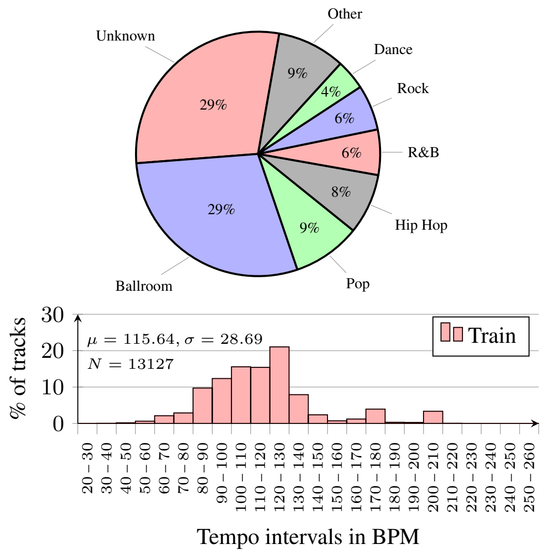

Figure 2. Distribution of genres and tempi for *Train*.

3.1.3 Temporal Flatness

To represent onset intensity we use a feature called *temporal flatness* (TF). Instead of calculating the Wiener entropy along the frequency axis of $Y(m, k)$, as we did for SF, we calculate it over a window of length ℓ along the time axis:

$$\text{TF}(m,\ell,k) = \frac{\left(\prod_{i=0}^{\ell-1} Y(m+i,k)\right)^{\frac{1}{\ell}}}{\frac{1}{\ell}\sum_{i=0}^{\ell-1} Y(m+i,k)} \qquad (5)$$

To compute TF values, we again re-use Y and limit k to $F(k) \in [30, 3000]$. Y is split into non-overlapping windows with length $\ell = 100$. For each bin k in a given window we compute TF. We then calculate the average $\text{TF}_\text{W}(m, \ell)$ over all k. As the two features for a song we use the mean and the variance of all its $\text{TF}_\text{W}(m, \ell)$ values.

3.2 Training Dataset

To avoid learning the test datasets, we use a dataset for training the classifier that has been created separately. *Train* is the union of an annotated, private music collection and the Extended Ballroom dataset [19] minus the 354 songs also occurring in the regular Ballroom set. Genre labels are available for 71% of the recordings. The genre as well as the tempo distribution are shown in Figure 2.

3.3 Correcting the Tempo Estimate

Differences between the training dataset's true tempo values and the estimated tempo values let us derive error class labels. With those and the proposed features, we can train a classifier. Using the classifier, we are then able to predict an estimated error class $E \in \mathbb{E}$ for any song for which we also have features and a tempo estimate. Note that this estimate does not have to stem from our own algorithm introduced in Section 2.1. One main idea of this paper, indeed,

Features	*Accuracy1*	*Accuracy2*
base	69.00-	**94.31**
LBS	75.84-	94.16
LBS + SF	76.68	94.09
LBS + TF	**77.41**	94.11
LBS + SF + TF	77.31	94.04
LBS_M	76.66	93.87
LBS_M + SF	75.91-	94.21
LBS_M + TF	77.31	93.97
LBS_M + SF + TF	76.21	94.14

Table 3. *Accuracy1* and *Accuracy2* for different feature combinations trained on *Train* and tested against *Combined*. The '−' signs indicate a statistically significant difference between the marked results and LBS + TF.

is that the classification model is algorithm-specific. In other words, the classifier must be trained for each tempo estimation algorithm. Once trained, it can be used to correct octave errors inherent to the given tempo estimation algorithm.

For the prediction process itself, we use a Random Forest [3] with 300 trees and a maximum depth of 25.

Given the estimated tempo T and the predicted error E the calculation of the corrected tempo $T_\text{corrected}$ is straight forward:

$$\begin{aligned} J(T,E) &= \begin{cases} 1 & \text{if } E = E_0 \\ i & \text{for } E_i \end{cases} \\ T_\text{corrected} &= T \cdot J(T,E) \end{aligned} \tag{6}$$

4. EVALUATION

In a first evaluation step, we compute *Accuracy1* for different feature combinations. We then compare the best combination with publicly available algorithms as well as other simple correction schemes.

4.1 Feature Evaluation

We trained the classifier using the dataset *Train* with different combinations of the proposed audio features and measured the performance against the dataset *Combined*. All tested feature combinations clearly outperformed the baseline algorithm base (T without correction) by at least 6pp (percentage points) for *Accuracy1* (Table 3). As was to be expected, *Accuracy2* didn't change significantly. The best performing feature combination was LBS + TF with an *Accuracy1* of 77.41%. When testing for significance with McNemar's test and a significance level of $p < 0.01$ [30], we found that LBS + TF performed significantly better w.r.t. *Accuracy1* than LBS, LBS_M + SF, and base. In the following we refer to the error classifier trained with LBS + TF as new. If no tempo estimation algorithm is explicitly mentioned, the method from Section 2.1 is otherwise implied.

4.2 Comparative Evaluation

We compared our method base+new to its baseline base and the three publicly available algorithms böck, stem, and schr using the test datasets described in Section 2.2.

böck [1] is the algorithm published by Böck et al. in [2], but trained with different datasets—among them our test data, i.e. the algorithm is "familiar" with the test sets. According to the authors, this configuration participated in MIREX 2016. stem is an algorithm aiming for low computational complexity published by Percival et al. [23]. We used the implementation contained in Marsyas 0.5.0. [2] Lastly, schr [3] was published by Schreiber et al. in [24]. Since our error estimation and correction method can be used as a post-processor for any tempo estimator, we also trained the classifier for each of these three tempo estimation algorithms to investigate potential improvements.

Figure 3 shows the *Accuracy1* results for the four algorithms when tested against *Combined*, both with and without new post-processing. All of them score significantly higher values when combined with new than in their plain form (McNemar, $p < 0.01$). The algorithms base (77.4%, increase of +8.4pp) and stem (74.7%, +5.3pp) clearly benefit the most, but also böck (74.5%, +2.1pp) and schr (76.6%, +3.9pp) gain several percent-

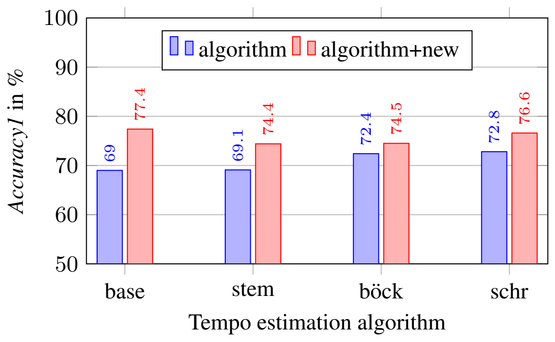

Figure 3. *Accuracy1* for *Combined* for different algorithms with and without new error correction. All algorithms reach significantly higher scores when combined with new.

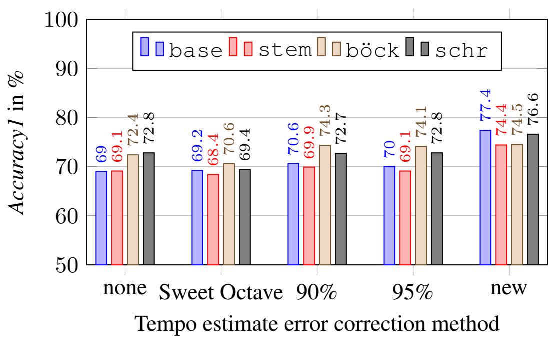

Figure 4. *Accuracy1* for *Combined* using no error correction, constraint-based correction, and new correction for various tempo estimation algorithms.

[1] https://github.com/CPJKU/madmom/

[2] http://marsyas.info/

[3] http://www.tagtraum.com/download/schreiber_icassp2014.zip

Dataset	`base`	`base+new`	`stem`	`stem+new`	`böck`	`böck+new`	`schr`	`schr+new`
ACM MIRUM	73.6	**81.8+**	74.3	79.9+	74.5	76.1+	76.3	81.3+
ISMIR04 Songs	65.5	69.2	60.1	62.3	55.4	58.4+	**74.1**	70.7-
Ballroom	64.3	85.1+	64.0	81.8+	84.8	**90.4+**	67.0	85.0+
Hainsworth	64.9	73.4+	69.8	74.8+	84.2	**85.6**	73.0	75.7
GTZAN	73.5	**78.8+**	77.9	76.9	70.7	71.1	78.0	76.2
SMC	45.2	39.6	29.5	29.5	51.1	**51.6**	41.5	35.5-
Dataset Average	64.5	71.3	62.6	67.5	70.1	**72.2**	68.3	70.7
Combined	69.0	**77.4+**	69.1	74.4+	72.4	74.5+	72.8	76.6+
GiantSteps	64.5	64.0	47.0	65.0+	61.5	**70.9+**	58.0	60.1
GiantSteps+Combined	68.4	**75.5+**	66.0	73.1+	70.8	74.0+	70.7	74.3+

Table 4. Tempo results for *Accuracy1* in percent. The '+' and '−' signs indicate a statistically significant difference between an algorithm and the same algorithm enhanced with `new`. Bold numbers mark the best-performing algorithm(s) for a dataset. *Dataset Average* is the mean of the algorithms' results for each dataset except GiantSteps.

age points.

As discussed in Section 2.3, a simple error correction scheme can be based on octave bias exploiting statistical properties of the test dataset. We therefore compared our method with such a constraint-based scheme, where tempi below a lower interval bound are doubled and above an upper bound are halved. For intervals we used the sweet octave and those listed in Table 2 with 90% and 95% coverage. Results are shown in Figure 4. Except for `böck`, none of the algorithms benefitted much from the simple correction—perhaps a certain bias is already built-in. When comparing `böck+new` and `böck+90%` we were not able to observe a significant difference. It appears, as if `böck`'s octave errors are harder to predict and correct than those of the other algorithms, perhaps because they are less systematic in nature.

Table 4 provides a detailed overview of *Accuracy1* results for each of the test algorithms for all test datasets. As mentioned, `base+new` reaches the highest score for the *Combined* dataset (77.4%). To the best of our knowledge, this is the highest *Accuracy1* score reported for *Combined* to date. For four of the six *Combined* datasets, `base+new` reaches significantly higher values than `base` (indicated by '+' signs in Table 4). The largest improvement was achieved for the Ballroom test set. The score for `base+new` is more than 20pp higher than `base`'s. The fact that 29% of *Train* consists of ballroom tracks certainly plays a role here. While the `base+new` score for ISMIR04 Songs is 3.7pp higher than `base`'s, the improvement is not significant. Similarly, the change for the SMC dataset (−5.6pp) is not significant, but noteworthy. We believe that both octave and genre bias may play a role here. Tracks in SMC are very different in style from those in *Train*. And compared to SMC, *Train* contains relatively few examples for slow tracks with 60 BPM or less. Informal tests confirm that choosing a different training dataset leads to better results.

Dataset-specific scores for `böck+new` are all higher than those for `böck`—more than half of them significantly. The largest increase can be observed for the GiantSteps dataset. Plain `böck` scores 61.5%—combined with `new` it reaches 70.9% (+9.4pp). To the best of our knowledge, this is the highest reported value for an unbiased, non-commercial algorithm to date.[4]

Dataset Average is the mean of the results for each of the six datasets in *Combined*. Because it is an unweighted average, it is not dominated by the larger datasets. But just like for *Combined*, we can observe higher scores for all algorithms when combined with `new`. With 72.2% (+2.1pp) `böck+new` reaches the highest score, closely followed by `base+new` with 71.3% (+6.8pp). `stem` (67.5%, +4.9pp) and `schr` (70.7%, +2.4pp) benefitted as well.

Though not the topic of this paper, we also measured *Accuracy2*. As expected, the results did not surprise and stayed stable.

5. CONCLUSIONS

We have shown that the proposed error correction method based on supervised learning of tempo estimation errors is capable of significantly improving *Accuracy1* results for existing tempo estimation algorithms. It does so in an algorithm-specific post-processing step. Combined with a simple tempo estimation algorithm, it outperforms other state-of-the-art algorithms for most of the tested datasets. We believe the error correction method can be enhanced even further by carefully selecting and incorporating other genre-related features.

We also discussed different kinds of biases that can have a large influence on the accuracy of tempo estimation algorithms. Ideally, evaluations of general purpose tempo estimators should be based on datasets with a mostly uniform tempo and genre distribution. Because better training data potentially leads to better results, training datasets should be an integral part of the comparison to make fair benchmarking possible. Defined train/test splits for existing datasets could be a first step in this direction.

Additional Material:
Binaries and other material are available at `http://www.tagtraum.com/tempo_estimation.html`.

Acknowledgments:
The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

[4] `http://www.cp.jku.at/datasets/giantsteps/`

6. REFERENCES

[1] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1047, 2005.

[2] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Accurate tempo estimation based on recurrent neural networks and resonating comb filters. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 625–631, Málaga, Spain, 2015.

[3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] Nick Collins. Towards a style-specific basis for computational beat tracking. In *Proceedings of the 9th International Conference on Music Perception and Cognition (ICMPC9) and 6th Triennial Conference of the European Society for the Cognitive Sciences of Music (ESCOM)*, Bologna, Italy, 2006.

[5] Anders Elowsson. Beat tracking with a cepstroid invariant neural network. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 351–357, New York, NY, USA, 2016.

[6] Anders Elowsson and Anders Friberg. Modeling the perception of tempo. *The Journal of the Acoustical Society of America*, 137(6):3163–3177, 2015.

[7] Jonathan Foote and Shingo Uchihashi. The beat spectrum: A new approach to rhythm analysis. In *Proceedings of the International Conference on Multimedia and Expo (ICME)*, Los Alamitos, CA, USA, 2001.

[8] Aggelos Gkiokas, Vassilios Katsouros, and George Carayannis. Reducing tempo octave errors by periodicity vector coding and svm learning. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 301–306. FEUP Edições, 2012.

[9] Fabien Gouyon, Anssi P. Klapuri, Simon Dixon, Miguel Alonso, George Tzanetakis, Christian Uhle, and Pedro Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1832–1844, 2006.

[10] Peter Grosche, Meinard Müller, and Frank Kurth. Cyclic tempogram – a mid-level tempo representation for music signals. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5522 – 5525, Dallas, Texas, USA, 2010.

[11] Peter Grosche, Meinard Müller, and Craig Stuart Sapp. What makes beat tracking difficult? A case study on Chopin Mazurkas. In *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, pages 649–654, Utrecht, The Netherlands, 2010.

[12] Stephen Webley Hainsworth. *Techniques for the Automated Analysis of Musical Audio*. PhD thesis, University of Cambridge, UK, September 2004.

[13] Jason Hockman and Ichiro Fujinaga. Fast vs slow: Learning tempo octaves from user data. In *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, pages 231–236, Utrecht, The Netherlands, 2010.

[14] Andre Holzapfel, Matthew EP Davies, José R Zapata, João Lobato Oliveira, and Fabien Gouyon. Selective sampling for beat tracking evaluation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(9):2539–2548, 2012.

[15] Florian Hörschläger, Richard Vogl, Sebastian Böck, and Peter Knees. Addressing tempo estimation octave errors in electronic music by incorporating style information extracted from wikipedia. In *Proceedings of the Sound and Music Computing Conference (SMC)*, Maynooth, Ireland, 2015.

[16] Anssi P. Klapuri. Sound onset detection by applying psychoacoustic knowledge. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3089–3092, Washington, DC, USA, 1999.

[17] Peter Knees, Ángel Faraldo, Perfecto Herrera, Richard Vogl, Sebastian Böck, Florian Hörschläger, and Mickael Le Goff. Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 364–370, Málaga, Spain, October 2015.

[18] Mark Levy. Improving perceptual tempo estimation with crowd-sourced annotations. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, pages 317–322, 2011.

[19] Ugo Marchand and Geoffroy Peeters. The extended ballroom dataset. In *Late Breaking Demo of the International Conference on Music Information Retrieval (ISMIR)*, New York, NY, USA, 2016.

[20] Meinard Müller. *Fundamentals of Music Processing – Audio, Analysis, Algorithms, Applications*. Springer Verlag, 2015.

[21] Geoffroy Peeters. Template-based estimation of time-varying tempo. *EURASIP Journal on Advances in Signal Processing*, 2007(1):158–158, 2007.

[22] Geoffroy Peeters and Joachim Flocon-Cholet. Perceptual tempo estimation using GMM-regression. In *Proceedings of the second international ACM workshop*

on Music information retrieval with user-centered and multimodal strategies, MIRUM '12, pages 45–50, New York, NY, USA, 2012. ACM.

[23] Graham Percival and George Tzanetakis. Streamlined tempo estimation based on autocorrelation and cross-correlation with pulses. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(12):1765–1776, 2014.

[24] Hendrik Schreiber and Meinard Müller. Exploiting global features for tempo octave correction. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 639–643, Florence, Italy, 2014.

[25] Björn Schuller, Florian Eyben, and Gerhard Rigoll. Tango or waltz?: Putting ballroom dance style into tempo detection. *EURASIP Journal on Audio, Speech, and Music Processing*, 2008:12, 2008.

[26] Jarno Seppänen. Tatum grid analysis of musical signals. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 131–134, 2001.

[27] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[28] George Tzanetakis and Graham Percival. An effective, simple tempo estimation method based on self-similarity and regularity. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, 2013.

[29] Fu-Hai Frank Wu and Jyh-Shing Roger Jang. A supervised learning method for tempo estimation of musical audio. In *Control and Automation (MED), 2014 22nd Mediterranean Conference of*, pages 599–604. IEEE, 2014.

[30] Jose R. Zapata and Emilia Gómez. Comparative evaluation and combination of audio tempo estimation approaches. In *42nd AES Conference on Semantic Audio*, Ilmenau, Germany, 2011.