JOHANNES KEPLER
UNIVERSITY LINZ

Submitted by
Florian Krebs

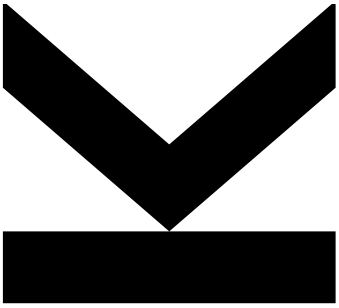
Submitted at
Department of Computational Perception

Supervisor and
First Examiner
Gerhard Widmer

Second Examiner
Geoffroy Peeters

November 2016

Metrical Analysis of Musical Audio Using Probabilistic Models



Doctoral Thesis
to obtain the academic degree of
Doktor der technischen Wissenschaften
in the Doctoral Program
Technische Wissenschaften

JOHANNES KEPLER
UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

Statutory Declaration

I hereby declare that the thesis submitted is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references.
This printed thesis is identical with the electronic version submitted.

Linz, December 21, 2016

Florian Krebs

Abstract

Due to the exploding amount of available music in recent years, media collections cannot be managed manually any more, which makes automatic audio analysis crucial for content-based search, organisation, and processing of data.

This thesis focuses on the automatic extraction of a metrical grid, determined by beats, downbeats, and time signature, from a music piece. I propose several algorithms to tackle this problem, all comprising three stages: First, (low-level) features are extracted from the audio signal. Second, an acoustic model transfers these features into probabilities in the music domain. Third, a probabilistic sequence model finds the most probable sequence of labels under the model assumptions.

This thesis provides contributions to the second and third stage. I (i) explore acoustic models based on machine learning methods, and (ii) develop models and algorithms for efficient probabilistic inference for both online and offline scenarios. Further, I design applications such as an automatic drummer which listens to and accompanies a musician in a live setting.

The most recent algorithms developed in this thesis exhibit state-of-the-art performance and clearly demonstrate the superiority of systems incorporating machine learning over hand-designed systems, which were prevalent at the time of starting this thesis. All algorithms developed in this thesis are publicly available as open-source software. I also publish beat and downbeat annotations for the Ballroom dataset to foster further research in this area.

Kurzfassung

In den letzten Jahren ist die Menge verfügbarer Musik explodiert. Um diese Datenmengen vernünftig organisieren und bearbeiten zu können, ist es essentiell, Methoden zu entwickeln die automatisch den Inhalt einer Mediendatei analysieren können.

Diese Dissertation beschäftigt sich mit der automatischen Analyse des musikalischen Metrums, einer Hierarchie von Pulsen unterschiedlicher Frequenzen. Zu diesem Zweck präsentiere ich mehrere Systeme, die alle aus drei grundlegenden Komponenten bestehen: Die erste Komponente extrahiert Merkmale aus dem Audiosignal. Diese Merkmale werden dann von der zweiten Komponente, dem akustischen Modell, in Wahrscheinlichkeiten in der musikalischen Domäne übersetzt. Die dritte Komponente besteht aus einem probabilistischen Modell, das die Wahrscheinlichkeiten in eine Sequenz von semantischen Labels übersetzt.

Diese Dissertation beschäftigt sich mit der zweiten und dritten Komponente, den akustischen und probabilistischen Modellen. Ich teste verschiedene Methoden des maschinellen Lernens auf ihre Eignung als akustisches Modell und entwickle Algorithmen die es erlauben effiziente Inferenz mit probabilistischen Modellen durchzuführen. Die beschriebenen Systeme werden dann in einen Schlagzeugroboter integriert, der selbstständig den Rhythmus eines Musikstückes analysieren und somit einen Musiker auf dem Schlagzeug begleiten kann.

Die präsentierten Methoden gehören zum aktuellen Stand der Technik im Bereich der automatischen metrischen Analyse von Musik. Die dargelegten Ergebnisse unterstreichen die Überlegenheit von maschinell gelernten Systemen gegenüber Systemen die hauptsächlich aus manuell gesetzten Regeln bestehen. Sowohl die entwickelten Algorithmen, als auch die im Laufe der Dissertation entstandenen Annotationen werden öffentlich zugänglich gemacht.

Acknowledgements

Writing a thesis is more than a single person working on a project for some time. In my case it has been a very special period of life, in a great city, in a great lab, and with great people. My first thanks go to Gerhard Widmer, who has brought me to Linz in the first place. He has been a great supervisor and invaluable advisor in my scientific career. Besides of being a great supervisor, he has managed to put together a group of very talented and open-minded people and to create the best conditions for a fruitful and enjoyable working atmosphere.

When I started working in the old TNF tower, I was welcomed warmly by my new co-workers with a daily breakfast of Weißwurst and chocolate croissants. Very soon I discovered the great potential of promising common projects within our working group: Maarten Grachten and I had wonderful music sessions in the lab's kitchen or on the roof of our ten-storey building, although the members of the neighbour institute did not always appreciate our interpretation of high-quality Gipsy art music. Another cooperation arose some months later, when Sebastian Böck, Andreas Arzt, Harald Frostel, and me participated in the University's championship in Bavarian Curling - and surprisingly won the first place. None of us had ever played this game before, but our endless team spirit gave us wings that day. These are only two little stories to illustrate the inspiring atmosphere at our department. For all the great moments I want to thank my friends and colleagues Markus Schedl, Peter Knees, Rainer Kelz, Filip Korzeniowski, Bruce Ferwerda, Maarten Grachten, Andreas Arzt, Sebastian Böck, Matthias Dorfer, Hamid Eghbal-Zadeh, Harald Frostel, Bernhard Lehner, Reinhard Sonnleitner, Andreu Vall, Richard Vogl, Tom Collins and Marko Tkalčič. Special thanks to Maarten Grachten and Sebastian Böck for being great companions in working and private hours, and introducing me to the secrets of Python. I also want to thank Filip Korzeniowski and Matthias Dorfer for all the Müsli, probabilistic models, Monte Carlo methods, and neural networks. I have learned a lot from you. And of course the best room mates ever, Andreas Arzt, Harald Frostel and Matthias Dorfer, for making sure working does not become too monotonous.

I want to thank Geoffroy Peeters for taking the time and effort to be reviewer and examiner of this thesis, and all the people from other departments that I had the honour of working with, especially Ajay Srinivasamurthy, Andre Holzapfel, Simon Durand,

and Matthew Davies. Thanks go also to Taylan Cemgil for his hospitable invitation to Boğaziçi University and for all the inspiring discussions about Bayesian methods.

Special thanks also to Andreu Vall, Filip Korzeniowski, and Richard Vogel, the members of the wonderful band *Caruso*, for all the rumba, punk and rock 'n' roll, for unforgettable musical moments between Linz and Alberndorf, and 9 great songs.

This work would not have been possible without the endless support of my family, who has always been there for me, despite the spatial distance. With all my heart I finally want to thank Caro who followed me to Linz and made the time in Linz the most productive one in my life. And thanks to my daughters Maya and Junah for showing me the magic of life every day in a new way.

Florian Krebs

November 2016

The research presented in this thesis was supported by the Austrian Science Fund (FWF) through the project Z159 (Wittgenstein Award).

Contents

List of abbreviations

CNN	Convolutional Neural Network
CRF	Conditional Random Field
DBN	Dynamic Bayesian Network
GMM	Gaussian Mixture Model
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
LSTM	Long Short-Term Memory
MCMC	Markov Chain Monte Carlo
ODF	Onset Detection Function
PF	Particle Filter
RNN	Recurrent Neural Network
SIS	Sequential Importance Sampling
STFT	Short-time Fourier Transform

Chapter 1

Introduction

1.1 Motivation and vision

The automatic extraction of rhythmic information from an audio signal is a research topic that has become more and more important over the last years. Due to the exploding amount of available audio data, current media collections cannot be managed manually any more, which makes it important for computers to 'understand' music in order to perform content-based search, organisation or processing of music. The goal of this thesis is to teach a computer to 'understand' the rhythmic structure in music, an ability that humans start to develop already in an early age (Winkler et al., 2009). I aim at developing a computer program that while listening to music, can identify the most salient levels of the metrical hierarchy, the beats and downbeats. To this end, a musical grammar mimicking the human perception has to be defined, in order to guide the computer's analysis. As rules and conventions for this musical grammar vary across different cultures (Stobart and Cross, 2000; Cameron et al., 2015) and are not always well defined (Parncutt, 1994), I want to use *machine learning* methods in order to learn these rules automatically from data. Music is also often ambiguous in its interpretation, therefore I believe that *probabilistic models* are a great tool to express this variability and ambiguity in a natural way.

1.2 Organisation

After giving background information on the content of this thesis in Chapter 2, I present my scientific contributions in Chapters 3-7, where each chapter represents one peer-reviewed publication. Small modifications to the papers were applied for formatting reasons, to make the appearance more uniform throughout the thesis, as well as in cases where errors in the original version were found. All modifications

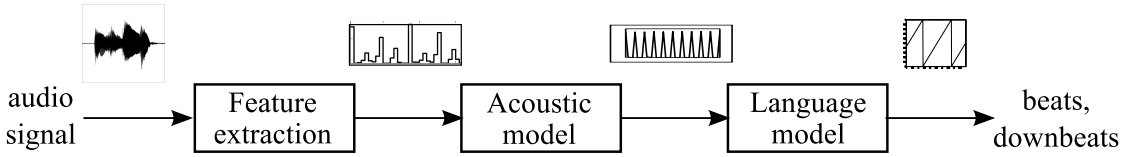


Figure 1.1: The three stages of a meter analysis system.

are mentioned at the beginning of each chapter, together with the co-authors and the conference/journal the publication appeared in. As all of the works were developed in team-work with other people, I also clarify my personal contributions at the beginning of each chapter. Chapter 8 describes how the proposed algorithms can be used to build an *online* rhythm analysis system. Chapter 9 shows applications, software toolboxes and datasets that have been developed in the course of this thesis. In Chapter 10, I finally draw conclusions and sketch ideas for future research.

1.3 Outline

As it happens often in scientific research, the path to the final state of this thesis has by no means been following a straight line. In this section, I illustrate how this path developed over the years by giving a chronological overview of my research and highlighting the connections between the various works.

The overall goal of this thesis is to develop a model that automatically performs a metrical analysis of a music piece, using only the audio signal. To tackle this problem, I propose several algorithms, all comprising three stages (see Fig. 1.1 for an overview): First, (low-level) *features are extracted* from the audio signal. Second, an *acoustic model* transfers these features into probabilities in the music domain. For example, this can be the probability that a certain audio frame contains a beat. Third, a probabilistic *language model* introduces prior musical knowledge and finds the most probable sequence of labels under the model assumptions. These labels are rhythmic descriptors like beats, downbeats, or time signature. The low-level feature extraction stage is described in Chapter 3 for the earlier systems, and in Chapter 7 for the most recent system. This thesis then mainly contributes to the development of the acoustic model (second stage; Chapters 3, 7) and the definition of the post-processing probabilistic model and inference methods (third stage; Chapters 4, 5, and 6).

In the following, I summarise each chapter and highlight the connections between them:

Chapter 3 I present a model for beat and downbeat tracking which explicitly models rhythmic patterns, in order to overcome shortcomings of existing methods. So far, rhythmic patterns for beat and downbeat detection had been either designed by

hand (Goto, 2001; Klapuri et al., 2006; Whiteley et al., 2006), or encompass only a single pattern (Peeters and Papadopoulos, 2011), mostly due to the lack of big representative datasets. For my work, I have annotated the Ballroom dataset (Section 2.4) with beat and downbeat times and use this data to learn rhythmic patterns from data, modelling each given dancestyle with one pattern. I extend the Dynamic Bayesian network (DBN) of Whiteley et al. (2006) by introducing a new acoustic model based on Gaussian mixture models (GMMs) and evaluate it on the Ballroom set. I show that in music with constant rhythm (like in Ballroom dance music), the incorporation of prior knowledge about rhythmic patterns improves the performance in downbeat tracking and reduces tempo octave errors in beat tracking.

Chapter 4 As a next step, I remove the dependency on given rhythmic pattern labels (e.g., dancestyle, as in chapter 3) for training the model. I present a training procedure, which learns the model parameters from *partially* annotated data (only a subset of the hidden variables are annotated), instead of requiring *all* hidden variables to be annotated. This allows using all available annotations for training the models, even if only a subset of the hidden variables are labeled. The annotations are converted into a 'plausibility' distribution over the hidden states and then fed into the subsequent Viterbi training stage to yield the model parameters. I use the proposed training method to find rhythmic patterns in music, for which beat and downbeat annotations are available, but rhythmic pattern labels are missing: First, the data is clustered via k-means, then the clusters are refined in several iterations of Viterbi training. With the proposed training method, I achieve a beat and downbeat tracking performance that is comparable to the case where pattern labels are provided by manual annotations.

Chapter 5 As the computational cost of exact inference in discrete-state DBNs grows with an increasing size of the state space, I investigate the application of approximate particle filter (PF) methods to perform efficient inference in the proposed meter analysis model. PFs are known to suffer from the so-called *degeneracy problem* (Doucet and Johansen, 2009), especially when dealing with long sequences and multi-modal probability distributions. These multi-modalities appear frequently in music due to its inherent ambiguity. To fight this degeneracy problem, I develop a method based on mixture PFs (Vermaak et al., 2003) which is able to track the most relevant modes in the posterior distribution over the state space throughout a music piece. In comparisons with a discrete-state DBN-based inference I achieve a beat and downbeat tracking accuracy that is only slightly lower at a drastically reduced computational cost.

Chapter 6 In this chapter, I revise the probabilistic model of Whiteley et al. (2006), which was the basis of all my previous publications. By modifying the state space and the way tempo transitions are implemented, I can halve the number of states of the

model and therefore can reduce the computation time by a factor up to 10 for discrete-state DBNs. At the same time, the beat and downbeat tracking performance is slightly improved. I test the proposed model with two observation models and on various musics, including Ballroom dance music, Pop/Rock music, and traditional Indian, Cretan and Turkish music.

Chapter 7 Due to the success of recurrent neural networks (RNNs) at detecting beats (Böck and Schedl, 2011), I explore RNNs for downbeat estimation. Given the beats, I aim at analysing the higher level of metrical structure, the downbeats and the time signature. Beat-synchronous features are processed by two parallel RNNs to compute a downbeat activation function. This function is then further post-processed by a DBN. In combination with a previously published beat tracker (Böck et al. (2014), Chapter 6), I am able to report state-of-the-art downbeat estimation results on seven datasets of Western music.

Chapter 8 In this chapter, I give technical details on how to make the so far described systems online-capable. I discuss several post-processing strategies and evaluate the beat tracking performance on three datasets. This material emerged while developing the applications described in Chapter 9 and has not been published.

Chapter 9 Together with students, I have created two prototype applications using the proposed meter analysis system: The first one is the *Automatic Dance Instructor*. It recognises dancestyle and position within a bar and synchronises a dance instruction video to the music. The second one is *Drumotron 3000*, a drum robot which listens to the music, analyses the meter, and accompanies a musician live. In this chapter, I shortly describe the systems and the graphical user interfaces. In addition, I introduce two software toolkits which contain most of the material of this thesis, as well as two datasets that have been generated for this thesis. Both software and datasets are publicly available.

1.4 Main contributions

Below, I summarise the main contributions of this thesis:

i) I propose two systems that infer beats, downbeats, and time signature from the audio signal of a music piece (Chapter 3). The parameters of the model can be learned automatically from annotated datasets, preventing potential biases induced by hand-crafted system components. I test the system on various musics, including Ballroom dance music (Chapter 3-7), Pop/Rock music (Chapter 4-7),

and traditional Indian, Cretan and Turkish music (Chapter 6-7, Holzapfel et al. (2014)). The systems are capable of both *online* and *offline* meter estimation.

Building on the model of Whiteley et al. (2006), I propose several fundamental improvements and extensions:

i) Acoustic models (observation models) using GMMs (Chapter 3) and RNNs (Chapter 7), both operating on low-level features derived from the audio signal.

ii) Reformulated discretisation of the state space and definition of the tempo transitions (Chapter 6).

iii) A training method that can make use of partially-labelled data, based on Viterbi training (Chapter 4).

iv) A scalable inference method that allows to track multiple modes of a posterior distribution by adapting mixture PFs to the problem of meter tracking (Chapter 5).

ii) I test the system in applications such as an automatic dance instructor, as well as a drum-playing robot.

iii) I publish the developed code (MATLAB and Python) to be used for further research (Chapter 9).

iv) I publish beat and downbeat annotations of the Ballroom dataset (Chapter 9), which comprises 685 audio excerpts with a total length of 5h 57m and currently 43 838 beat annotations.

1.5 MIREX evaluation results

Since 2005, the *International Music Information Retrieval Systems Evaluation Laboratory* (IMIRSEL) at the *Graduate School of Library and Information Science* (GSLIS), *University of Illinois at Urbana-Champaign* (UIUC) organises a yearly evaluation of music information retrieval (MIR) systems, which is called MIREX (*MIR Evaluation eXchange*). Participants submit their systems, which are then evaluated on (mostly) non-public data on the servers of the UIUC. I have submitted most algorithms of this thesis to the MIREX evaluation tasks *Audio Beat Tracking* and *Audio Downbeat Estimation*. The best results (F-measure metric (FM)) of our systems are shown in Tables 1.1 - 1.2. For other metrics please consult the MIREX webpage [1].

In the task *Audio Beat Tracking*, there have been about 100 submissions to the *Audio Beat Tracking* task since its first taking place in 2006, although the number of unique

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

participating teams is significantly lower (One team usually submits several versions of a system). The algorithms developed in this thesis achieved several first places throughout the years (see Table 1.1). In the MCK dataset, our system BK1 (Chapter 6) performs only 0.3% worse than the current (2016) best system, a difference which is not statistically significant. In the SMC dataset, which contains especially difficult excerpts for beat tracking, our systems obtained first and second rank in 2012. After 2012, our systems still are among the best performing ones, but the results are not reported here as the RNNs were trained on the SMC dataset as well and thus might be over-fitted.

In the task *Audio Downbeat Estimation*, I have counted 20 submissions to the *Audio Downbeat Estimation* task in the period between 2014 and 2016. The submitted systems perform best in five of eight datasets (see Table 1.2). Results for the three missing datasets are not listed here, as the beat tracker was trained on these sets and thus a comparison with other systems is not fair. More realistic results for these datasets can thus be found in our publications (Holzapfel et al. (2014), Chapter 7).

Dataset	Year	System	FM	Rank
MCK	2012	FK1 (Chapter 3)	56.7 (56.7)	1
MCK	2014	BK6 (Böck et al., 2014)	61.3 (61.3)	1
MCK	2015-2016	BK1 (Chapter 6)	63.6 (63.9)	3
SMC	2012	KB1 (Krebs and Böck, 2012)	40.7 (40.7)	1
SMC	2012	FK1 (Chapter 3)	39.7 (40.7)	2
MAZ	2012	FK1 (Chapter 3)	58.4 (66.6)	2

Table 1.1: MIREX results *Audio Beat Tracking*; The results in parentheses (in column "FM") give the performance of the best system up to the corresponding year. FM is the beat tracking F-measure.

Dataset	Year	System	FM	Rank
Carnatic	2014	KSH1 (Holzapfel et al. (2014), Chapter 3)	40.0 (40.0)	1
Carnatic	2015-2016	FK4 (Chapter 6)	47.4 (47.4)	1
Cretan	2014-2016	FK3 (Chapter 4)	53.5 (53.5)	1
HJDB	2015-2016	FK3 (Chapter 6)	82.4 (82.4)	1
RWC classical	2016	KB1 (Chapter 7)	43.6 (43.6)	1
GTZAN	2016	KB2 (Chapter 7)	64.7 (64.7)	1

Table 1.2: MIREX results *Audio Downbeat Estimation*; The results in parentheses (in column "FM") give the performance of the best system up to the corresponding year. FM is the downbeat tracking F-measure.

Chapter 2

Background

2.1 Definition of musical terms

In this section, I will define some terms that are used throughout this thesis.

Meter Musical meter is defined by a hierarchy of pulses, which occur at integer-related frequencies. These different pulses can also be referred to as *metrical levels* (Lerdahl and Jackendoff, 1987). Three of these metrical levels are of particular interest (see Fig. 2.1 for an illustration): (i) The *tatum*, originating from 'temporal atom' (Bilmes, 1993), is the lowest metrical level and is associated with the highest pulse frequency. (ii) The *beat*, is rather loosely-defined as the pulse that humans perceive as being the most salient and choose to tap their feet to. Its frequency is an integer fraction of the tatum rate. (iii) The *downbeat*, *measure* or *bar* pulse has a frequency which is an integer fraction of the beat rate, and is related to the time signature of a piece (and hence the length (in beats) of a musical bar). In the example shown in Fig. 2.1 the frequency ratio between tatum and beat is two, and between beat and downbeat frequency is four. Bar lines indicate the beginning of a bar and define the downbeats (which happen to coincide with a musical event only in the first of the three bars in this particular example).

metrical levels

tatum

beat

downbeat

The integer ratio between the beat frequency and the next lower level determines the type of the meter: In *simple meters* (e.g., $2/4$, $3/4$, $4/4$), the beat interval is divided into two groups (the next lower metrical level has a frequency which is twice the beat frequency), whereas in *compound meters* (e.g., $6/8$, $9/8$), each beat is divided into three groups (the next lower metrical level has a frequency that is three times the beat frequency). As beats and downbeats are a perceptional construct, listeners differ in their interpretation of meter, caused by individual and cultural factors (Stobart and Cross, 2000).

simple meter

compound meter

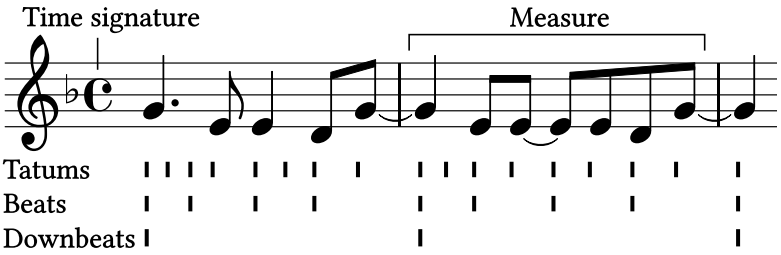


Figure 2.1: Illustration of tatums, beats and downbeats in a musical score

Time signature The time signature (only used in Western music notation) defines both the length of a measure in terms of note values (e.g., quarter notes or eighth notes) as well as the number of beats per measure. E.g., a time signature of $3/4$ means that a measure contains three beats, and each beat lasts one quarter note. The convention of representing time signatures as a fraction of two numbers can be confusing in the case of compound meters: E.g., a time signature of $6/8$ means that the measure lasts sixth eighth notes, but the most salient pulse period (the beat period) does not necessarily have to be an eighth note long. In most $6/8$ measures, the beat period is a dotted quarter note, meaning that there are only two beats per measure. In the score shown in Fig. 2.1 the time signature is denoted by the 'common time' symbol after the clef, which stands for $4/4$.

Tempo The tempo of a music piece is the frequency of the beat pulse, measured in beats per minute.

2.2 Dynamic Bayesian Networks

Data is often the observable result of a temporal process and comes in the form of sequences, e.g., the stock price, audio signals, or a recording of human brain activity. Music is obviously a sequential phenomenon as well, because the single notes of a music piece do not make sense without their sequential ordering. In the following, I will introduce a class of sequential probabilistic models called *dynamic Bayesian networks* (DBNs), which I will use to model musical meter.

2.2.1 Definition

In a (latent) *state-space model*, it is assumed that the data sequences (also called observations or measurements) are generated by some underlying hidden state of the world, and that this hidden state evolves in time. Dynamic Bayesian Networks (DBNs) (see Murphy (2002) for a good overview) are a powerful class of state-space models which

represent these 'states' and 'observations' by random variables and provide algorithms for training and probabilistic inference. For example, Hidden Markov Models (HMMs) are a popular special case of DBNs, where the hidden states consist only of a *single* random variable (which is mostly discrete). DBNs extend HMMs by modelling *sets* of both discrete and continuous random variables. In the following, I will denote the hidden states as x_k^i where k is the time index, and i is the index of a specific random variable. DBNs can be represented by a graph where the nodes represent random variables and

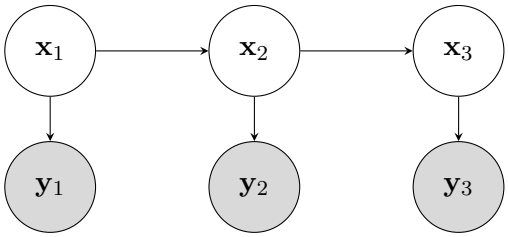


Figure 2.2: Dynamic Bayesian network; The gray nodes are observed, and the white nodes represent the hidden variables.

the edges represent dependencies between the variables. Fig. 2.2 shows an example of such a graph. A DBN is defined by the conditional probability distributions (CPDs) of each node given its parents. To make inference tractable in DBNs, three assumptions are generally made (see also Fig. 2.2):

1. The hidden states at time k depend only on the hidden states of time $k-1$. Then, the model becomes first-order Markov (*Markov assumption*).

2. Observations at time k depend only on the hidden states at time k. This means, given the hidden states, the observations are conditionally independent.

3. Both $P(\mathbf{x}_k|\mathbf{x}_{k-1})$ (the *transition model*) and $P(\mathbf{y}_k|\mathbf{x}_k)$ (the *observation model*) are supposed to be the same for all time steps k (*Stationarity assumption*).

In our example, this means the model is defined by specifying $P(\mathbf{x}_1)$, $P(\mathbf{x}_k|\mathbf{x}_{k-1})$, and $P(\mathbf{y}_k|\mathbf{x}_k)$. Exploiting the three assumptions mentioned above, the *joint distribution* over the hidden states $\mathbf{x}_k = [x_k^1, ..., x_k^{N_h}]$ and the observations $\mathbf{y}_k = [y_k^1, ..., y_k^{N_o}]$ can be written compactly as

joint distribution

$$P(\mathbf{x}_{1:K}, \mathbf{y}_{1:K}) = P(\mathbf{x}_1, \mathbf{y}_1) \prod_{k=2}^{K} P(\mathbf{x}_k|\mathbf{x}_{k-1}) P(\mathbf{y}_k|\mathbf{x}_k). \tag{2.1}$$

2.2.2 Inference

Often, we are interested in inferring the hidden states at each time point from a set of observations. This corresponds to computing or maximising the posterior $P(\mathbf{x}|\mathbf{y})$ and is described in the following, for online and offline scenarios.

Online inference

Filtering

For online applications, we want to compute $P(\mathbf{x}_k|\mathbf{y}_{1:k})$, the distribution over the hidden states given only observations that have happened until the current time k. This can be computed recursively by

$$P(\mathbf{x}_k|\mathbf{y}_{1:k}) \propto P(\mathbf{y}_k|\mathbf{x}_k) \int_{\mathbf{x}_{k-1}} P(\mathbf{x}_k|\mathbf{x}_{k-1}) P(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1}). \tag{2.2}$$

In case of discrete distributions, the integral in Eq. 2.2 can be replaced by a sum. We use *proportional* instead of *equal* in Eq. 2.2 since the distribution would have to be divided by $P(\mathbf{y}_k|\mathbf{y}_{1:k-1})$ in order to be a valid probability distribution. With discrete probability distributions, this is trivial as it corresponds to simply normalising the distribution by its sum. Most of the times, however, we are interested in simply finding the most probable state and do not care about its actual probability, so the normalisation constant can be neglected. An example of how the most probable hidden state can be selected from the filtering distribution is given in Chapter 8 for an online beat tracking scenario.

Offline inference

MAP estimate

For offline applications, we are mostly interested in finding the sequence of hidden states $\mathbf{x}_{1:K}$ that maximises the posterior probability distribution $P(\mathbf{x}_{1:K}|\mathbf{y}_{1:K})$, given the whole sequence of observations. This is often called the *MAP estimate* (maximum a-posterior) estimate of a sequence of hidden states $\mathbf{x}^*_{1:K}$ and is computed by

$$\mathbf{x}^*_{1:K} = \arg\max_{\mathbf{x}_{1:K}} P(\mathbf{x}_{1:K} \mid \mathbf{y}_{1:K}). \tag{2.3}$$

MAP estimates of models with discrete hidden state variables can be efficiently computed by the Viterbi algorithm (Viterbi, 1967) and will be used in Chapters 3-7.

Exact Inference

Kalman filter

Exact inference according to Equations 2.2 and 2.3 is only possible under certain conditions. One example is the Kalman Filter (KF) model (Kalman, 1960; Roweis and Ghahramani, 1999). It assumes that all random variables are continuous-valued and all distributions (e.g., the transition and observation model) are from the exponential family,

such as the Gaussian distribution. These distributions have the property 'closure under multiplication', which means computing the integral in Equation 2.2 only changes the parameters of the distributions, but does not make them more complex (Minka, 1999). E.g., if both transition and observation densities are Gaussian, the filtering density is guaranteed to stay Gaussian as well.

grid-based methods

Another example are grid-based methods, where the state space is divided into cells (Arulampalam et al., 2002). This produces a discrete state space where inference can be executed exactly, by replacing all integrals by summations. Obviously, in the case of continuous state spaces, a discretisation always means a loss of accuracy and hence the solutions are not 'exact' anymore. One example of this class are HMMs which have been very popular in the speech recognition community. Like in KF models, there exist efficient algorithms for solving Equation 2.2 (Forward algorithm) and Equation 2.3 (Viterbi algorithm) (Rabiner, 1989).

HMMs

Approximate Inference

In cases where the above-mentioned conditions do not hold (e.g., continuous random variables with distributions outside the exponential family, or mixed discrete / continuous state spaces), an approximate solution can still be found. One possibility is to use *variational inference*, where the complex distributions of the model are approximated by simpler variational distributions, but this is outside the scope of this thesis. Another popular class of algorithms are *Monte Carlo* (MC) methods, which represent a continuous distribution by a set of samples ('particles') that are drawn from this distribution. If the number of particles is large, it can be assumed that the approximation is sufficient. This can be seen as another discretisation of the state space, but using a dynamic grid instead of a fixed grid. The problem with MC methods is that it is usually difficult to sample from the distribution of interest (e.g., the filtering distribution). In these cases, Markov Chain Monte Carlo (MCMC) or Particle filters (PF) can be applied to obtain samples that approximate any target distribution. A more detailed description of PF methods can be found in Section 5.4.2 or in Arulampalam et al. (2002); Doucet and Johansen (2009).

Monte Carlo methods

2.2.3 Learning

Learning in DBNs is a wide topic and has many dimensions as sketched by Murphy (2002). In this thesis, I concentrate on learning the *parameters* of a model, but do not learn the *structure* of a model. Learning the structure of a model is computationally expensive and would need a huge amount of training data. Therefore, I decided to engineer the structure by hand, and use it as a means of incorporating prior musical knowledge into the model.

The parameters of a DBN can be learned either in a completely unsupervised way (when no information about the hidden states is available), in a completely supervised way (when we have a dataset where all the hidden states are labeled), or by a mixture of both cases (when only a fraction of the hidden variables are labeled). In the completely supervised case, learning becomes trivial and mainly reduces to counting the occurrences of each hidden state and converting them to probabilities. In contrast, in the completely unsupervised case, learning usually employs the iterative *Expectation-Maximisation* (EM) algorithm (Dempster et al., 1977) and is therefore more complex. An alternative to the EM algorithm is *Viterbi training* or *segmental k-means* (Rabiner et al., 1986) which instead of computing the forward-backward path (Rabiner, 1989) only computes the Viterbi path at each iteration, and is therefore faster and computationally less expensive. For the purpose of this thesis, I will either learn the parameters from fully-annotated data (Chapters 3, 5-7) or use a variation of Viterbi training that allows to learn the model parameters from partially-labeled data (Chapter 4).

2.3 Previous work on probabilistic meter analysis

In recent years, there has been a growing interest in solving the problem of automatic metrical analysis of music. In this section, I give a short review of relevant literature for this thesis, especially publications on meter analysis with probabilistic models. For a more detailed, publication-specific literature review I refer the reader to the introductions of the corresponding chapters.

Table 2.1 lists meter analysis systems based on probabilistic models.[1] One of the advantages of probabilistic models is that dependencies between variables can easily be modelled by designing the structure of the model. As can be seen from Table 2.1, most systems simultaneously infer multiple hidden variables. This has the advantage of being able to exploit their mutual dependencies, but has the drawback of increasing the state space and therefore the computational cost to do inference. E.g., Whiteley et al. (2006) jointly infer tempo, beat phase, measure length and measure phase. In practice, such a state space becomes huge and makes inference intractable, therefore approximate inference schemes such as particle filters (PFs) or Markov Chain Monte Carlo (MCMC) are often employed.

Systems can be categorised according to their input features, the acoustic model, and the type of language model.

[1] The list does not claim to be complete. It rather tries to cover a wide range of different approaches.

system	acoustic model	language model	joint inference			
			tempo	beat phase	measure length	measure phase
Beat estimation						
Cemgil et al. (2000)	tempogram	Kalman filter	x	x		
Cemgil and Kappen (2003)	onsets	Monte Carlo methods (PFs and MCMC)	x	x		
Laroche (2003)	beat template	dynamic programming	x	x		
Hainsworth and Macleod (2004)	onset detector	PF	x	x		
Lang and de Freitas (2004)	beat template	PF / discrete DBN	x	x		
Eck (2007)	autocorrelation phase matrix	discrete DBN	x	x		
Degara et al. (2011)	complex flux	HMM				
Böck et al. (2014)	RNN	discrete DBN	x	x		
Fillon et al. (2015)	tempogram, beat template	CRF	x	x		
Beat and downbeat estimation						
Klapuri et al. (2006)	comb filters	discrete DBN	x		x	
Whiteley et al. (2006)	Gaussian process / Poisson model	discrete DBN	x	x	x	x
Whiteley et al. (2007)	Poisson model	PFs	x	x	x	x
Papadopoulos and Peeters (2011)	distance to chord template	discrete DBN			x	x
Peeters and Papadopoulos (2011)	bar template, spectral balance, chroma change	discrete DBN		x	x	x
Khadkevich et al. (2012)	spectral flux, chroma variation	HMMs	x	x	x	x
Krebs et al. (2015b)	GMM	PF	x	x	x	x
Böck et al. (2016b)	RNN	discrete DBN	x	x	x	x
Krebs et al. (2016)	RNN	discrete DBN			x	x
Holzapfel and Grill (2016)	CNN	discrete DBN	x	x	x	x
Downbeat estimation						
Durand et al. (2016)	CNN	discrete DBN			x	x

Table 2.1: Comparison of meter analysis systems based on probabilistic models.

Input features The first approaches to meter analysis used discrete onset times as input. These were either derived from MIDI recordings (Cemgil et al., 2000; Cemgil and Kappen, 2003), or were computed by an onset detector (Hainsworth and Macleod, 2004; Lang and de Freitas, 2004). Now, continuous features are usually extracted from the audio stream. The most common ones are variations of the *Spectral Flux* with one frequency band (Laroche, 2003; Eck, 2007; Degara et al., 2011; Peeters and Papadopoulos, 2011; Khadkevich et al., 2012; Fillon et al., 2015), two frequency bands (Krebs et al., 2015a,b), three frequency bands (Durand et al., 2016), four frequency bands (Klapuri et al., 2006), and more than four bands (Böck et al., 2016b; Krebs et al., 2016). For downbeat detection, chroma features (Papadopoulos and Peeters, 2011; Peeters and Papadopoulos, 2011; Khadkevich et al., 2012; Durand et al., 2016; Krebs et al., 2016) are popular features to detect harmonic change which often happens at the beginning of a bar. In recent years, automatic feature learning from raw (filtered) spectrograms has become feasible and popular (Böck and Schedl, 2011; Böck et al., 2016b) and the resulting features have been shown to outperform hand-crafted features, as demonstrated in the yearly MIREX evaluation of beat tracking systems[2]. Spectrogram-based features have also been adopted in my most recent work on downbeat tracking (Chapter 7).

Acoustic model The acoustic model converts the input features to observation probabilities. The correlation between a beat template and the spectral flux is used directly as observation probability by Laroche (2003); Peeters and Papadopoulos (2011); Fillon et al. (2015). The value of the (complex) flux is used directly as observation probability by Degara et al. (2011). The strengths of periodicities in the input features are used as observation probabilities by Cemgil et al. (2000); Klapuri et al. (2006); Eck (2007); Fillon et al. (2015). Recently, due to the increasing computing power and amount of available data, systems contain complex acoustic models, whose numerous parameters are learned from data. These models are usually more powerful and can easily be adapted to any style by providing annotated training data. Examples of such acoustic models encompass my own work using GMMs (Chapters 3-6) and RNNs (Chapters 6-7, Böck et al. (2016b)), as well as work by Durand et al. (2015) and Holzapfel and Grill (2016) using convolutional neural networks (CNNs).

Language model The language model implements the dynamics of the model and converts a sequence of observation probabilities into a sequence of musical parameters (beats, downbeats, time signature, etc.). Cemgil et al. (2000) use a Kalman filter to detect beats given a sequence of onset times. Another popular model family are particle filters (PF) (Cemgil and Kappen, 2003; Hainsworth and Macleod, 2004; Lang and de Freitas, 2004; Sethares et al., 2005; Whiteley et al., 2007; Krebs et al., 2015b), which, in contrast to Kalman filters, do not place high demands on the type of state variables (continuous,

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

discrete, mixed) and probability distributions. A third approach is to discretise the (naturally continuous-valued) variables and use HMMs (Degara et al., 2011; Peeters and Papadopoulos, 2011; Khadkevich et al., 2012) or, in the case of several hidden variables, discrete DBNs (Klapuri et al., 2006; Whiteley et al., 2006; Eck, 2007; Papadopoulos and Peeters, 2011; Krebs et al., 2015a; Durand et al., 2016; Krebs et al., 2016). Furthermore, undirected probabilistic models such as conditional random fields (CRFs) have been proposed for beat tracking by Lang and de Freitas (2004); Korzeniowski et al. (2014); Fillon et al. (2015).

The methods in this thesis are all based on DBNs, which provide an intuitive and elegant way to model the dependencies between multiple random variables, such as tempo, the position in a bar, and time signature. I cover DBNs consisting only of discrete states (Chapters 3-7), as well as DBNs which consist of a mixture of continuous and discrete states (PF models, Chapter 5). While the discrete DBNs are usually computationally more complex than the PF models, they often provide higher accuracy and have the advantage of being deterministic.

2.4 Datasets

For development and evaluation of the proposed methods, various datasets have been used in this thesis. Table 2.2 lists the datasets and some of their characteristics.

Dataset	Reference	# pieces	Length	Annotations	Genre
Ballroom	Gouyon et al. (2006); Krebs et al. (2013);	685 [3]	5h 57	beats, downbeats	mixed
Beatles	Davies et al. (2009)	180	8h 09	beats, downbeats	rock
Carnatic_118[4]	Srinivasamurthy and Serra (2014)	118	3h 55	beats, downbeats	Carnatic music
Cretan	Holzapfel et al. (2014)	42	2h 20	beats, downbeats	Cretan dances
Hainsworth	Hainsworth and Macleod (2004)	222	3h 19	beats, downbeats[5]	mixed
HJDB	Hockman et al. (2012)	235	3h 19	beats, downbeats	Electronic dance music
Klapuri	Klapuri et al. (2006)	320	4h 54	downbeats	mixed
RWC Pop	Goto et al. (2002)	100	6h 47	beats, downbeats	rock
Robbie Williams	Giorgi et al. (2013)	65	4h 31	beats, downbeats	rock
Rock	De Clercq and Temperley (2011)	200	12h 53	beats, downbeats	rock
SMC	Holzapfel et al. (2012)	217	2h 25	beats	mixed
Turkish / Usul[6]	Srinivasamurthy et al. (2014)	82	1h 22	beats, downbeats	Turkish music
1360-song[7]	Gouyon (2005)	1360	11h 02	beats	mixed

Table 2.2: Datsets used in this thesis.

[3]Duplicates have been removed as pointed out by Sturm (2014).
[4]This is a subset of the original Carnatic dataset, which has 176 full-length pieces.
[5]Downbeat annotations have been added later by Sebastian Böck.
[6]This is a subset of the original Turkish dataset, which has 93 full-length pieces.
[7]This is a collection of several other datasets, including SIMAC, Hainsworth, Cuidado, and Klapuri.

Chapter 3

Rhythmic Pattern Modeling for Beat and Downbeat Tracking in Musical Audio

Published In Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR) (Krebs et al., 2013).

Authors Florian Krebs, Sebastian Böck, and Gerhard Widmer

Personal contributions I did all the implementations and ran the experiments. Sebastian later integrated my MATLAB code into the *madmom* (Böck et al., 2016a) framework written in Python and assisted me in annotating the Ballroom dataset.

Changes to the original paper I have modified the results section of the original paper in order to make it consistent with the rest of this thesis: First, for the evaluation, we do not skip any pauses at the beginning of a song. Second, we show here the same evaluation metrics that are used in the rest of this thesis: F-measure, CMLt, and AMLt for beat tracking, and F-measure for downbeat tracking. Third, we leave out 13 replicated songs in the Ballroom dataset which were identified by Sturm (2014). Finally, in Section 3.3.4, I corrected a typo in the footnote: We use eight components for the GMM with PS8 instead of four as stated in the original paper.

Abstract Rhythmic patterns are an important structural element in music. This paper investigates the use of rhythmic pattern modeling to infer metrical structure in musical audio recordings. We present a Hidden Markov Model (HMM) based system that simultaneously extracts beats, downbeats, tempo, meter, and rhythmic patterns.

Our model builds upon the basic structure proposed by Whiteley et al. (2006), which we further modified by introducing a new observation model: rhythmic patterns are learned directly from data, which makes the model adaptable to the rhythmical structure of any kind of music. For learning rhythmic patterns and evaluating beat and downbeat tracking, 685 ballroom dance pieces were annotated with beat and measure information. The results show that explicitly modeling rhythmic patterns of dance styles drastically reduces octave errors (detection of half or double tempo) and substantially improves downbeat tracking.

3.1 Introduction

From its very beginnings, music has been built on temporal structure to which humans can synchronize via musical instruments and dance. The most prominent layer of this temporal structure (which most people tap their feet to) contains the approximately equally spaced *beats*. These beats can, in turn, be grouped into *measures*, segments with a constant number of beats; the first beat in each measure, which usually carries the strongest accent within the measure, is called the *downbeat*. The automatic analysis of this temporal structure in a music piece has been an active research field since the 1970s and is of prime importance for many applications such as music transcription, automatic accompaniment, expressive performance analysis, music similarity estimation, and music segmentation. However, many problems within the automatic analysis of metrical structure remain unsolved. In particular, complex rhythmic phenomena such as syncopations, triplets, and swing make it difficult to find the correct phase and period of downbeats and beats, especially for systems that rely on the assumption that beats usually occur at onset times. Considering all these rhythmic peculiarities, a general model no longer suffices.

One way to overcome this problem is to incorporate higher-level musical knowledge into the system. For example, Hockman et al. (2012), proposed a genre-specific beat tracking system designed specifically for the genres hardcore, jungle, and drum and bass. Another way to make the model more specific is to model explicitly one or several *rhythmic patterns*. These rhythmic patterns describe the distribution of note onsets within a predefined time interval, e.g., one bar. For example, Goto (2001) extracts bar-length drum patterns from audio signals and matches them to eight prestored patterns typically used in popular music. Klapuri et al. (2006) proposed a HMM representing a three-level metrical grid consisting of tatum, tactus, and measure. Two rhythmic patterns were employed to obtain an observation probability for the phase of the measure pulse. The system of Whiteley et al. (2006) jointly models tempo, meter, and rhythmic patterns in a Bayesian framework. Simple observation models were proposed for symbolic and audio data, but were not evaluated on polyphonic audio signals.

Although rhythmic patterns are used in some systems, no systematic study exists that investigates the importance of rhythmic patterns for analyzing the metrical structure. Apart from the approach presented by Peeters and Papadopoulos (2011), which learns a single rhythmic template from data, rhythmic patterns to be used for beat tracking have so far only been designed by hand and hence depend heavily on the intuition of the developer.

This paper investigates the role of rhythmic patterns in analyzing the metrical structure in musical audio signals. We propose a new observation model for the HMM-based system described by Whiteley et al. (2006), whose parameters are learned from real audio data and can therefore be adapted easily to represent any rhythmic style.

3.2 Rhythmic Patterns

Although rhythmic patterns could be defined at any level of the metrical structure, we restrict the definition of rhythmic patterns to the length of a single measure.

3.2.1 Data

As stated in Section 3.1, strong deviations from a straight on-beat rhythm constitute potential problems for automatic rhythmic description systems. While pop and rock music is commonly concentrated on the beat, Afro-Cuban rhythms frequently contain syncopations, for instance in the *clave* pattern – the structural core of many Afro-Cuban rhythms. Therefore, Latin music represents a serious challenge to beat and downbeat tracking systems.

The ballroom dataset[1] contains eight different dance styles (Cha cha, Jive, Quickstep, Rumba, Samba, Tango, Viennese Waltz, and (slow) Waltz) and has been used by several authors, for example, for genre recognition (Dixon et al., 2004; Pohle et al., 2009). It consists of 685 unique (Sturm, 2014) 30 seconds-long audio excerpts and has tempo and dance style annotations. The dataset contains two different meters (3/4 and 4/4) and all pieces have constant meter. The tempo distributions of the dance styles are displayed in Fig. 3.4.

We have annotated both beat and downbeat times manually. In cases of disagreement on the metrical level we relied on the existing tempo and meter annotations. The annotations can be downloaded from `https://github.com/CPJKU/BallroomAnnotations`.

3.2.2 Representation of rhythmic patterns

Patterns such as those shown in Fig. 1 are learned in the process of inducing the likelihood function for the model (cf. Section 3.3.3), where we use the dance style labels of

[1]The data was extracted from `www.ballroomdancers.com`.

the training songs as indicators of different rhythmic patterns. To model dependencies between instruments in our pattern representations, we split the audio signal into two frequency bands and compute an onset feature for each of the bands individually as described in Section 3.3.3. To illustrate the rhythmic characteristics of different dance styles, we show the eight learned representations of rhythmic patterns in Fig. 3.1. Each pattern is represented by a distribution of onset feature values along a bar in two frequency bands.

For example, the *Jive* pattern displays strong accents on the second and fourth beat, a phenomenon usually referred to as *backbeat*. In addition, the typical *swing* style is clearly visible in the high-frequency band. The *Rumba* pattern contains a strong accent of the bass on the 4th and 7th eighth note, which is a common bass pattern in Afro-Cuban music and referred to as *anticipated bass* (Manuel, 1985). One of the characteristics of *Samba* is the shuffled bass line, a pattern originally played with the *Surdo*, a large Brazilian bass drum. The pattern features bass notes on the 1st, 4th, 5th, 9th, 12th, and 13th sixteenth note of the bar. *Waltz*, finally, is a triple meter rhythm. While the bass notes are located mainly on the downbeat, high-frequency note onsets are also located at the quarter and eighth note level of the measure.

3.3 Method

In this section, we describe the *dynamic Bayesian network* (DBN) (Murphy, 2002) we use to analyze the metrical structure. We assume that a time series of *observed* data $\mathbf{y}_{1:K} = \{\mathbf{y}_1, ..., \mathbf{y}_K\}$ is generated by a set of unknown, *hidden* variables $\mathbf{x}_{1:K} = \{\mathbf{x}_1, ..., \mathbf{x}_K\}$, where K is the length of an audio excerpt in frames. In a DBN, the joint distribution $P(\mathbf{y}_{1:K}, \mathbf{x}_{1:K})$ factorizes as

$$P(\mathbf{y}_{1:K}, \mathbf{x}_{1:K}) = P(\mathbf{x}_1) \prod_{k=2}^{K} P(\mathbf{x}_k|\mathbf{x}_{k-1}) P(\mathbf{y}_k|\mathbf{x}_k) \tag{3.1}$$

where $P(\mathbf{x}_1)$ is the *initial state distribution*, $P(\mathbf{x}_k|\mathbf{x}_{k-1})$ is the *transition model*, and $P(\mathbf{y}_k|\mathbf{x}_k)$ is the *observation model*.

The proposed model is similar to the model proposed by Whiteley et al. (2006) with the following modifications:

- We assume that the tempo depends on the rhythmic pattern (cf., Section 3.3.2), which is a valid assumption for ballroom music as shown in Fig. 3.4.
- As the original observation model was mainly intended for percussive sounds, we replace it by a Gaussian Mixture Model (GMM) as described in Section 3.3.3.

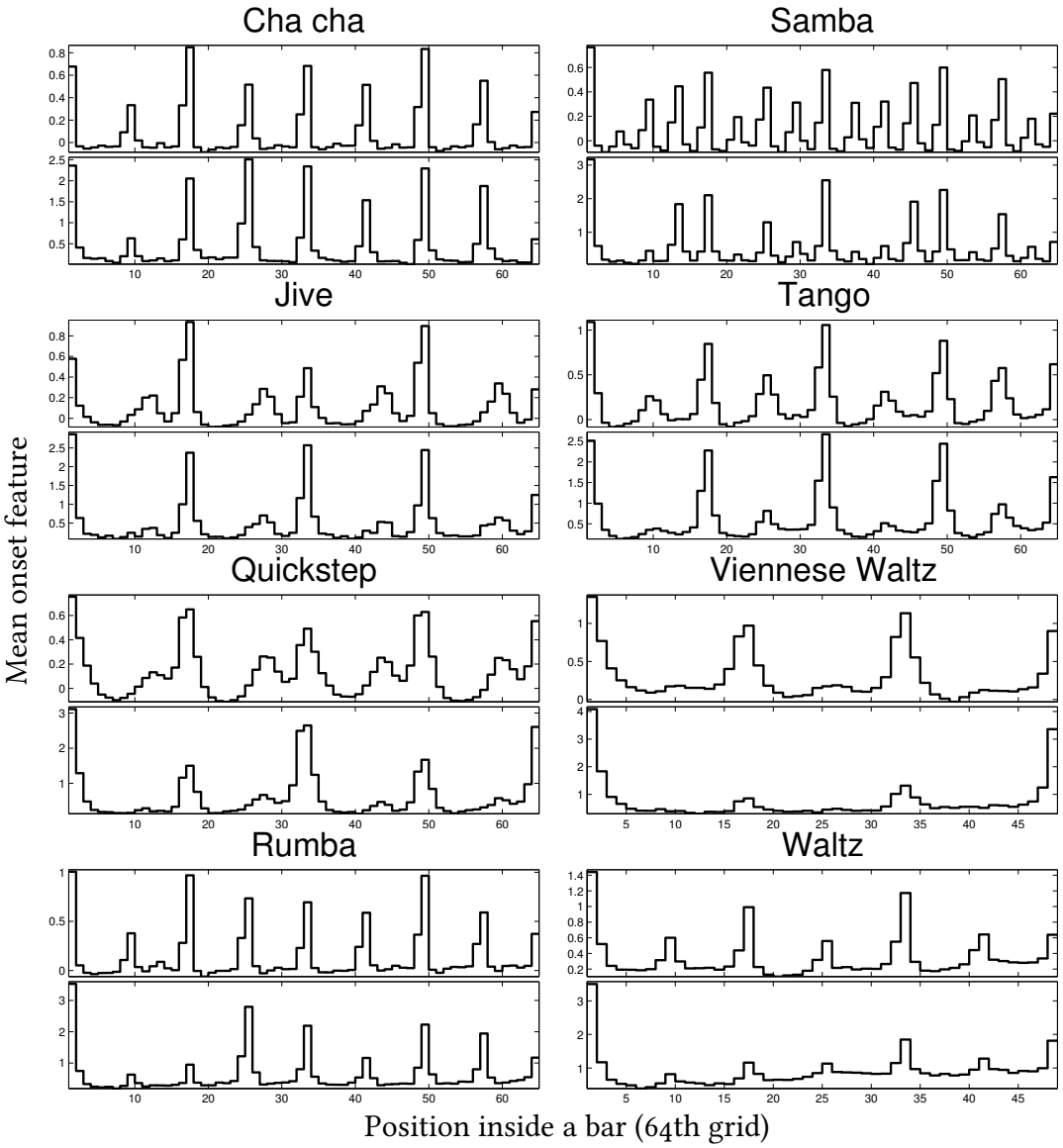


Figure 3.1: Illustration of learned rhythmic patterns. For each pattern, two frequency bands are shown (Low/High from bottom to top).

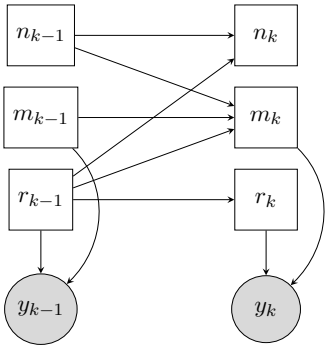


Figure 3.2: Dynamic Bayesian network; circles denote continuous variables and rectangles discrete variables. The gray nodes are observed, and the white nodes represent the hidden variables.

3.3.1 Hidden variables

The *dynamic bar pointer model* (Whiteley et al., 2006) defines the state of a hypothetical bar pointer at time $t_k = k \cdot \Delta$, with $k \in \{1, 2, ..., K\}$ and Δ the audio frame length, by the following discrete hidden variables:

1. Position inside a bar $m_k \in \{1, 2, ..., M\}$, where
   $m_k = 1$ indicates the beginning and $m_k = M$ the end of a bar;

2. Tempo $n_k \in \{1, 2, ..., N\}$ (unit $\frac{\text{bar positions}}{\text{audio frame}}$), where N denotes the number of tempo states;

3. Rhythmic pattern $r_k \in \{r_1, r_2, ..., r_R\}$, where R denotes the number of rhythmic patterns.

For the experiments reported in this paper, we chose $\Delta = 20\,\text{ms}$, $M = 1216$, $N = 26$, and R (the number of rhythmic patterns) was 2 or 8 as described in Section 3.4.2. Furthermore, each rhythmic pattern is assigned to a meter $\theta(r_k)$
$\in \{3/4, 4/4\}$, which is important to determine the measure boundaries in Eq. 3.4. The conditional independence relations between these variables are shown in Fig. 3.2.

As noted by Murphy (2002), any discrete state DBN can be converted into a regular HMM by merging all hidden variables of one time slice into a 'meta-variable' $\mathbf{x}_k$, whose state space is the Cartesian product of the single variables:

$$\mathbf{x}_k = [m_k, n_k, r_k]. \tag{3.2}$$

3.3.2 Transition model

Due to the conditional independence relations shown in Fig. 3.2, the transition model factorizes as

$$\begin{aligned} P(\mathbf{x}_k|\mathbf{x}_{k-1}) = \quad & P(m_k|m_{k-1}, n_{k-1}, r_{k-1})\times \\ & \times P(n_k|n_{k-1}, r_{k-1}) \times P(r_k|r_{k-1}) \end{aligned} \tag{3.3}$$

where the three factors are defined as follows:

- $P(m_k|m_{k-1}, n_{k-1}, r_{k-1})$
  At time frame k the bar pointer moves from position m_{k-1} to m_k as defined by

$$m_k = [(m_{k-1} + n_{k-1} - 1)\text{mod}(N_m \cdot \theta(r_{k-1}))] + 1. \tag{3.4}$$

  Whenever the bar pointer crosses a bar border it is reset to 1 (as modeled by the modulo operator).

- $P(n_k|n_{k-1}, r_{k-1})$
  If the tempo n_{k-1} is inside the allowed tempo range $\{n_{min}(r_{k-1}), ..., n_{max}(r_{k-1})\}$, there are three possible transitions: the bar pointer remains at the same tempo, accelerates, or decelerates:
  if $n_{min}(r_{k-1}) \leq n_{k-1} \geq n_{max}(r_{k-1})$,

$$P(n_k|n_{k-1}) = \begin{cases} 1 - p_n, & n_k = n_{k-1}; \\ \frac{p_n}{2}, & n_k = n_{k-1} + 1; \\ \frac{p_n}{2}, & n_k = n_{k-1} - 1. \end{cases} \tag{3.5}$$

  Transitions to tempi outside the allowed range are assigned a zero probability. p_n is the probability of a change in tempo per audio frame, and the step-size of a tempo change per audio frame was set to one bar position per audio frame.

- $P(r_k|r_{k-1})$
  For this work, we assume a musical piece to have a characteristic rhythmic pattern that remains constant throughout the song; thus we obtain

$$r_{k+1} = r_k. \tag{3.6}$$

3.3.3 Observation model

For simplicity, we omit the frame indices k in this section. The observation model $P(\mathbf{y}|\mathbf{x})$ reduces to $P(\mathbf{y}|m, r)$ due to the independence assumptions shown in Fig. 3.2.

Observation features

Since the perception of beats depends heavily on the perception of played musical notes, we believe that a good onset feature is also a good beat tracking feature. Therefore, we use a variant of the *LogFiltSpecFlux* onset feature, which performed well in recent comparisons of onset detection functions (Böck et al., 2012b) and is summarized in the top part of Fig. 3.3. We believe that the bass instruments play an important role in defining rhythmic patterns, hence we compute onsets in low-frequencies (< 250 Hz) and high-frequencies (> 250 Hz) separately. In Section 3.5.1 we investigate the importance of using the two-dimensional onset feature over a one-dimensional one. Finally, we subtract the moving average computed over a window of one second and normalize the features of each excerpt to zero mean and unity variance.

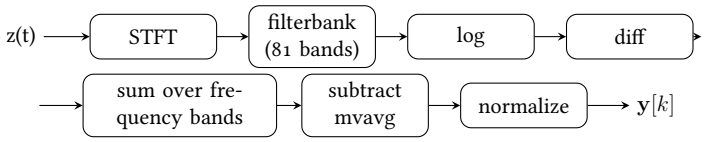


Figure 3.3: Computing the onset feature $\mathbf{y}[\mathrm{k}]$ from the audio signal $z(t)$

State tying

We assume the observation probabilities to be constant within a 64th note grid. All states within this grid are tied and thus share the same parameters, which yields 64 (4/4 meter) and 48 (3/4 meter) different observation probabilities per bar and rhythmic pattern.

Likelihood function

To learn a representation of $P(\mathbf{y}|m, r)$, we split the training dataset into pieces of one bar length, starting at the downbeat. For each bar position within the 64th grid and each rhythmic pattern, we collect all corresponding feature values and fit a GMM. We achieved the best results on our test set with a GMM of $I = 2$ components. Hence, the observation probability is modeled by

$$P(\mathbf{y}|m, r) = \sum_{i=1}^{I} w_{m,r,i} \cdot \mathcal{N}(\mathbf{y}; \mu_{m,r,i}, \Sigma_{m,r,i}), \tag{3.7}$$

where $\mu_{m,r,i}$ is the mean vector, $\Sigma_{m,r,i}$ is the covariance matrix, and $w_{m,r,i}$ is the mixture weight of component i of the GMM. Since, in learning the likelihood function $P(\mathbf{y}|m, r)$, a GMM is fitted to the audio features for every rhythmic pattern (i.e., dance

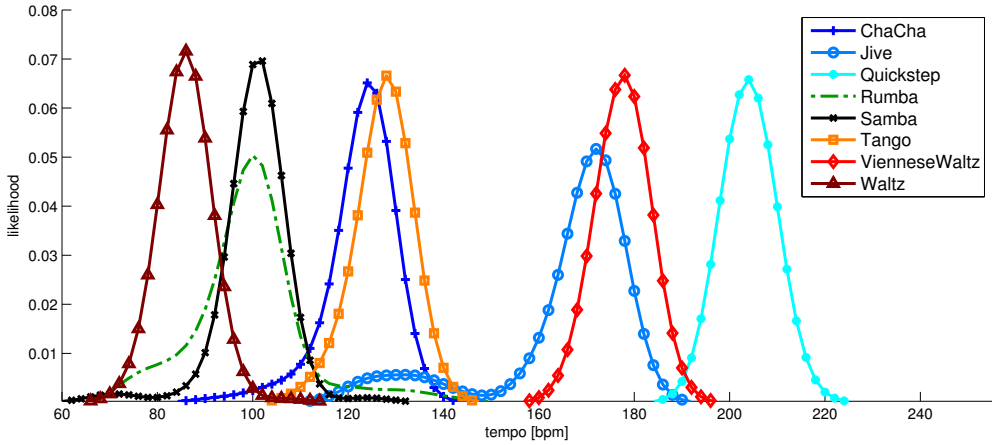


Figure 3.4: Tempo distributions of the ballroom dataset dance styles. The displayed distributions are obtained by (Gaussian) kernel density estimation for each dance style separately.

style) label r, the resulting GMMs can be interpreted directly as representations of rhythmic patterns. Fig. 1 shows the mean values of the features per frequency band and bar position for the GMMs corresponding to the eight rhythmic patterns $r \in$ {Cha cha, Jive, Quickstep, Rumba, Samba, Tango, Viennese Waltz, Waltz}.

3.3.4 Initial state distribution

The bar position and the rhythmic patterns are assumed to be distributed uniformly, whereas the tempo state probabilities are modeled by fitting a GMM[2] to the tempo distribution of each ballroom style shown in Fig. 3.4.

3.3.5 Inference

We are looking for the state sequence $\mathbf{x}^*_{1:K}$ with the highest posterior probability $p(\mathbf{x}_{1:K}|\mathbf{y}_{1:K})$:

$$\mathbf{x}^*_{1:K} = \arg\max_{x_{1:K}} p(\mathbf{x}_{1:K}|\mathbf{y}_{1:K}). \tag{3.8}$$

We solve Eq. 3.8 using the Viterbi algorithm (Viterbi, 1967; Rabiner, 1989). Once $\mathbf{x}^*_{1:K}$ is computed, the set of beat and downbeat times are obtained by interpolating $m^*_{1:K}$ at the corresponding bar positions.

[2]We use two (PS2), and eight (PS8) mixture components repectively.

3.4 Experimental setup

We use different settings and reference methods to evaluate the relevance of rhythmic pattern modeling for the beat and downbeat tracking performance.

3.4.1 Evaluation measures

A variety of measures for evaluating beat tracking performance is available (Davies et al., 2009). Based on their popularity and relevance for our experiments we chose to report the metrics F-measure, CMLt, and AMLt for beat tracking as well as F-measure for downbeat tracking:

- FM (F-measure) is computed over a commonly used detection window of +/- 70 ms around a beat/downbeat annotation.
- CMLt (Correct Metrical Level with no continuity required) assesses the percentage of correct beats at the correct metrical level.
- AMLt (Allowed Metrical Level with no continuity required) assesses the percentage of correct beats, also considering half and double tempo and offbeats as correct.

All metrics are used with standard settings (Davies et al., 2009), except that we do not exclude the first five seconds of an excerpt from the evaluation. Due to lack of space, we present only the mean values per measure across all files of the dataset. Please visit `http://www.cp.jku.at/people/krebs/ISMIR2013.html` for detailed results and other metrics.

3.4.2 Systems compared

To evaluate the use of modeling multiple rhythmic patterns, we report results for the following variants of the proposed system (PS): PS2 uses two rhythmic patterns (one for each meter), PS8 uses eight rhythmic patterns (one for each genre), PS8.genre has the ground truth genre, and PS2.meter has the ground truth meter as additional input features.

In order to compare the system to the state-of-the-art, we add results of six reference beat tracking algorithms: Ellis (2007), Davies and Plumbley (2007), Degara et al. (2011), Böck and Schedl (2011), Peeters and Papadopoulos (2011), and Klapuri et al. (2006). The latter two also output downbeat times.

3.4.3 Parameter training

For all variants of the proposed system PSx, the results were computed by a leave-one-out approach, where we trained the model on all songs except the one to be tested. System of Böck and Schedl (2011) has been trained on the data specified in their paper, the SMC (Holzapfel et al., 2012), and the Hainsworth dataset (Hainsworth and Macleod,

2004). The beat templates used by Peeters and Papadopoulos (2011) have been trained using their own annotated PopRock dataset. The other methods do not require any training.

3.4.4 Statistical tests

In Section 3.5.1 we use an analysis of variance test (ANOVA) and in Section 3.5.2 a multiple comparison test (Hochberg and Tamhane, 1987) to find statistically significant differences among the mean performances of the different systems. A significance level of 0.05 was used to declare performance differences as statistically relevant.

3.5 Results and discussion

3.5.1 Dimensionality of the observation feature

As described in Section 3.3.3, the onset feature is computed for one (PS*x*.1d) or two (PS*x*.2d) frequency bands separately. The top parts of Table 3.1 show the effect of the dimensionality of the feature vector on the beat and downbeat tracking results respectively.

For beat tracking, analyzing the onset function in two separate frequency bands seems to help finding the correct metrical level, as indicated by higher CMLt measures in Table 3.1. Even though the improvement is not significant, this effect was observed for both PS2 and PS8.

For downbeat tracking, we have found a significant improvement if two bands are used instead of a single one, as evident from the rightmost column of Table 3.1. This seems plausible, as the bass plays a major role in defining a rhythmic pattern (see Section 3.2.2) and helps to resolve the ambiguity between the different beat positions within a bar.

Using three or more onset frequency bands did not improve the performance further in our experiments. In the following sections we will only report the results for the two-dimensional onset feature (PS*x*.2d) and simply denote it as PS*x*.

3.5.2 Relevance of rhythmic pattern modeling

In this section, we evaluate the relevance of rhythmic pattern modeling by comparing the beat and downbeat tracking performance of the proposed systems to six reference systems.

System	FM	CMLt	AMLt	DBFM
PS2.1d	**80.4**	**62.0**	**87.5**	56.6
PS2.2d	**81.8**	**65.7**	**87.4**	**64.0**
PS8.1d	**84.3**	**75.2**	**86.8**	65.8
PS8.2d	**85.4**	**78.3**	**86.3**	**70.8**
PS2	81.8	65.7	**87.4**	64.0
PS8	**85.4**	**78.3**	**86.3**	**70.8**
Ellis (2007)	68.2	30.9	79.1	-
Davies and Plumbley (2007)	76.1	56.9	**86.2**	-
Degara et al. (2011)	78.9	62.9	84.5	-
Peeters and Papadopoulos (2011)	76.3	56.7	84.1	42.1
Böck and Schedl (2011)	81.6	63.1	**88.9**	-
Klapuri et al. (2006)	72.8	53.7	81.5	43.3
PS2.meter	82.5	67.4	88.1	67.9
PS8.genre	90.4	88.8	89.6	78.5

Table 3.1: Beat and downbeat detection performance on the ballroom dataset. Results printed in bold are statistically equivalent to the best result in each experiment. FM, CMLt, and AMLt are beat tracking metrics, DBFM denotes downbeat tracking F-measure.

Beat tracking

The beat tracking results of the reference methods are displayed together with PS2 (=PS2.2d) and PS8 (=PS8.2d) in the middle part of Table 3.1. As there is no single system that performs best in all of the measures, we will look at the individual metrics in the following.

For the CMLt and FM metric, PS8 outperforms all other systems. This is especially clear for the CMLt metric (which requires the system to exactly report the annotated metrical level), where PS8 achieves a relative improvement of 24% over the best reference system.

For the AMLt metric (which also allows detecting beats at half or double tempo and offbeat), we found no advantage of using the proposed methods over most of the reference methods. Böck and Schedl (2011) performs best in this metric, even though the difference to PS2, PS8 and Davies and Plumbley (2007) is not significant.

From the fact that the proposed model PS8 performs best if finding the correct tempo octave is required (as in CMLt), we conclude that modeling rhythm patterns seems to be beneficial for choosing the correct metrical level. This gets even clearer if the correct dance style is supplied (PS8.genre). In this case, the CMLt score is almost identical to the AMLt score. Apparently, the dance style provides sufficient rhythmic information to resolve ambiguities in choosing the metrical level.

Hence, if the correct metrical level is unimportant or even ambiguous, a general model like Böck and Schedl (2011) or any other reference system might be preferable to the more complex and specific PS8. On the contrary, in applications where the correct metrical level matters (e.g., a system that detects beats and downbeats for automatic ballroom dance instructions (Eyben et al., 2007)), PS8 is the best system to chose.

Knowing the meter a priori (PS2.meter) was not found to increase the performance significantly compared to PS2. It appeared that meter was identified mostly correct by PS2 (in 89% of the songs) and that for the remaining 11% songs both of the rhythmic patterns fitted equally well.

Downbeat tracking

The rightmost column of Table 3.1 lists the results for downbeat tracking. As shown, PS8 outperforms all other systems significantly. In cases where the dance style is known a priori (PS8.genre), the downbeat performance increases even more. The same was observed for PS2 if the meter was known (PS2.meter). This leads to the thought that downbeat tracking (as well as beat tracking with PS8) could improve even more by including meter or genre detection methods. For instance, Pohle et al. (2009) report a dance style classification rate of 89% on the same dataset, whereas PS8 detected the correct dance style in only 75% of the cases.

The poor performance of the systems Peeters and Papadopoulos (2011) and Klapuri et al. (2006) is probably caused by the fact that both systems were developed for music with a completely different metrical structure than present in ballroom data. In addition, Klapuri et al. (2006) explicitly assumes a 4/4 meter, which is only true for 522 of 685 songs in the dataset.

3.6 Conclusion and future work

In this study, we investigated the influence of explicit modeling of rhythmic patterns on the beat and downbeat tracking performance in musical audio signals. For this purpose we have proposed a new observation model for the system proposed by Whiteley et al. (2006), representing rhythmical patterns in two frequency bands.

Our experiments indicated that computing an onset feature for at least two different frequency bands increases the downbeat tracking performance significantly compared to a single feature covering the whole frequency range.

In a comparison with six reference systems, explicitly modeling dance styles as rhythmic patterns was shown to drastically reduce octave errors (detecting half or double tempo) in beat tracking. Besides, downbeat tracking was improved substantially compared to a variant that only models meter and two reference systems.

Obviously, ballroom music is well structured in terms of rhythmic patterns and

tempo distribution. If all the findings reported in this paper also apply to music genres other than ballroom music has yet to be investigated.

In this work, the rhythmic patterns were determined by dance style labels. In future work, we want to use unsupervised clustering methods to extract meaningful rhythmic patterns from the audio features directly.

Chapter 4

Unsupervised Learning and Refinement of Rhythmic Patterns

Published In Proceedings of the 22nd European Signal Processing Conference (EUSIPCO) (Krebs et al., 2014).

Authors Florian Krebs, Filip Korzeniowski, Maarten Grachten, and Gerhard Widmer.

Personal contributions In discussions with Filip we developed the idea to generalise our previous work towards unsupervised learning of rhythmic patterns using Viterbi training. I did most of the implementation work.

Changes to the original paper

1. I have modified the results section of the original paper in order to make it consistent with the rest of this thesis: We report the same evaluation metrics that are used in the rest of this thesis: F-measure, CMLt, and AMLt for beat tracking, and F-measure for downbeat tracking.

2. Again, I leave out 13 replicated songs in the Ballroom dataset as identified by Sturm (2014).

3. As I found several errors in the beat annotations of the Hainsworth dataset after the paper was published, I decided to re-evaluate the algorithms' performance using the new annotations.

4. In the original paper we conducted a fixed number of iterations of Viterbi training, wheras in this thesis I report results for the iteration that yields the best beat tracking score on a validation set.

5. Instead of showing results for 3 and 8 patterns on the Hainsworth set, I present results for 2, 5, and 10 patterns now, as this more clearly illustrates the relationship between the performance and the number of patterns.

All these changes did not affect the main conclusions of the paper, but did slightly change the results.

Abstract In this paper, we propose a method of extracting rhythmic patterns from audio recordings to be used for training a probabilistic model for beat and downbeat extraction. The method comprises two stages: clustering and refinement. It is able to take advantage of any available annotations that are related to the metrical structure (e.g., beats, tempo, downbeats, dance style). Our evaluation on the Ballroom dataset showed that our unsupervised method achieves results comparable to those of a supervised model. On another dataset, the proposed method performs comparable to the reference systems, except for one beat tracking metric.

4.1 Introduction

A musical work can be regarded as a composition of sounds in time. Most musical works are *metrical*, which means that the time at which sounds occur is organised into a hierarchical structure consisting of multiple pulse levels. The perceptually most prominent level is referred to as the *beat*, which coincides with the pulse most people would naturally tap their foot to. These beats are in turn organised into groups of equal numbers of beats (*measures*), with the *downbeat* denoting the first and strongest beat of each measure. The distribution of sound events within the metrical structure is referred to as *rhythm*.

Automatic analysis of the rhythmic structure of a musical work from an audio recording facilitates many aspects of music information retrieval research, such as music transcription, music similarity estimation, and music segmentation. The great challenge when designing an automatic rhythm transcription system lies in the great rhythmic variety of music. A general rhythm extraction system should work for all metres; it should handle swing, syncopations, tuplets as well as different tempo ranges. It is therefore unlikely that simple approaches such as rule-based methods can satisfy these requirements.

Whiteley et al. (2006) proposed a system that jointly models bar position, tempo, metre, and rhythmic pattern using a hidden Markov model (HMM). Recently, Krebs et al. (2013) have extended this model by incorporating eight rhythmic pattern states

to model the eight dance styles in the data. We showed that it outperforms state-of-the-art beat and downbeat detection systems on a dataset of 697 ballroom music excerpts. However, the parameters of these rhythmic pattern states are learned in a supervised manner: the system needs a list of rhythmic pattern labels as input.

In the present paper we describe an approach to learning the model parameters (see Section 4.2) in a semi-unsupervised manner, using beat and downbeat annotated data but no rhythmic pattern labels. Thus, rhythmic pattern labels are no longer necessary. The proposed method is as follows: After clustering the bars of the dataset based on onset features (Section 4.3.1), we refine the parameter estimation using Viterbi training (Section 4.3.2). Finally, we apply the system to a dataset for which rhythmic pattern labels are not available (Section 4.4).

4.2 Model description

In this section, we describe the beat and downbeat detection system used in this work. For a more detailed description, the interested reader is referred to Krebs et al. (2013).

We take features extracted from the audio signal as observed variables $\mathbf{y}_k$, and infer the hidden variables $\mathbf{x}_k$ of bar position, tempo, and rhythmic pattern on the basis of a hidden Markov model (HMM). Here, k denotes the audio frame index with $1 \leq k \leq K$, where K is the number of audio frames extracted for a piece. In an HMM, the joint probability distribution of a sequence of hidden states $\mathbf{x}_{1:K}$ and observed states $\mathbf{y}_{1:K}$ factorises as

$$P(\mathbf{x}_{1:K}, \mathbf{y}_{1:K}) = P(\mathbf{x}_1) \prod_{k=2}^{K} P(\mathbf{x}_k|\mathbf{x}_{k-1}) P(\mathbf{y}_k|\mathbf{x}_k), \tag{4.1}$$

where $P(\mathbf{x}_1)$ is the initial distribution of the hidden states, $P(\mathbf{x}_k|\mathbf{x}_{k-1})$ is the transition model, and $P(\mathbf{y}_k|\mathbf{x}_k)$ is the observation model. The state space of the hidden variables $\mathbf{x}_k$ is the Cartesian product of three discrete sub-state spaces: the position inside a bar $m \in \{1, 2, ..., M\}$, the tempo $n \in \{1, 2, ..., N\}$, and the rhythmic pattern $r \in \{r_1, r_2, ..., r_R\}$.[1] Thus, a state at audio frame k in this state space is written as $\mathbf{x}_k = [m_k, n_k, r_k]$.

The sequence of hidden states with the maximum a posteriori probability $\mathbf{x}^*_{1:K}$ is computed using the Viterbi algorithm as

$$\mathbf{x}^*_{1:K} = \arg\max_{\mathbf{x}_{1:K}} \{ P(\mathbf{x}_{1:K} \mid \mathbf{y}_{1:K}, \lambda) \}, \tag{4.2}$$

where λ are the parameters of the model.

[1] In this paper, we used $M = 1216$, $N = 23$.

4.2.1 Transition model

To make inference in this large state space computationally feasible, the system is restricted to a limited number of transitions per state. We allow for three possible tempo state transitions, as modeled by the transition probabilities for n: if $n_k \in \{n_{min}(r_k), ..., n_{max}(r_k)\}$,

$$P(n_k|n_{k-1}) = \begin{cases} 1 - p_n, & n_k = n_{k-1}, \\ \frac{p_n}{2}, & n_k = n_{k-1} + 1, \\ \frac{p_n}{2}, & n_k = n_{k-1} - 1, \end{cases} \tag{4.3}$$

where p_n is the probability of a tempo change, and $n_{min}(r_k)$ and $n_{max}(r_k)$ are respectively the minimum and maximum tempo corresponding to the rhythmic pattern r_k. Transitions to tempi outside the allowed range are assigned a zero probability.

The bar position m_k at time frame k is defined deterministically by the previous bar position m_{k-1} and the previous tempo n_{k-1}.

Finally, the rhythmic pattern state is assumed to change only at bar boundaries:

$$P(r_k|r_{k-1}, m_k < m_{k-1}) = p_r(r_{k-1}, r_k). \tag{4.4}$$

The transition probabilities of the rhythmic pattern states $p_r(r_{k-1}, r_k)$, the tempo transition probability p_n, and the allowed tempo ranges $n_{min}(r_k)$ and $n_{max}(r_k)$ are learned from data as described in Section 4.3.

4.2.2 Observation model

The observed variables $\mathbf{y_k}$ are computed using the *LogFiltSpecFlux* method introduced by Böck et al. (2012b), calculated for two frequency bands (below 250 Hz and above 250 Hz). The observation likelihood $P(\mathbf{y}_k|m_k, r_k)$ is modeled by a Gaussian mixture model (GMM) with two components. In order to obtain a manageable number of parameters to learn, the observation probabilities were assumed to be constant for the duration of a 64^{th} note.

4.3 Learning

The model's rigid structure reduces the set of state transition parameters to $n_{min}(r_k)$, $n_{max}(r_k)$, p_n, and $p_r(r_{k-1}, r_k)$. Learning thus mainly focuses on extracting rhythmic patterns in form of the observation likelihood $P(\mathbf{y}_k \mid m_k, r_k)$ from the data. We obtain all parameters in a two-phase process: First, a simple k-means approach gives us an initial model; then, we refine this model via multiple runs of the segmental k-means algorithm (Rabiner et al., 1986), also called *Viterbi training*.

We chose the parameter estimation methods based on the available data (see Section 4.4.1 for details). If we had fully annotated data samples, we could simply determine all transition probabilities by counting, and learn the observation model by a single maximum likelihood estimation. However, our training data is only partially labelled: We have annotated beat and downbeat times, but no information on the bar positions between those annotations[2]. Additionally, manual beat annotations tend to be imprecise, as mentioned by Dixon et al. (2006). Therefore, we can determine neither rhythmic patterns nor transitions directly.

We briefly describe the initialisation process in Section 4.3.1 before detailing the model refinement in Section 4.3.2.

4.3.1 K-Means Initialisation

Algorithms for learning the parameters of a HMM usually modify a given initial model towards a specified goal by minimising an objective function. While a random initial model will suffice in some cases, an informed initialisation often provides the optimiser with a better starting point, resulting in faster convergence. In our system, initial values for tempo transitions are set by hand ($p_n = 0.01$). Parameters concerning rhythmic patterns, however, are determined from training data.

Rhythmic patterns define the observation likelihood $P(\mathbf{y}_k \mid m_k, r_k)$ as described in Section 4.2.2. Given many degrees of freedom, finding sensible initial values is crucial. To compute these, we apply the following steps:

1. Calculate the frequency split *LogFiltSpecFlux* for each audio file in the training corpus.

2. Split the feature values into single bars based on the ground truth annotation.

3. Group the features for each bar into 64 bins equally spaced in time within each bar. This results in 128 values per bar, since we compute the onset feature for two distinct frequency bands. Determined by the ground truth, for metres other than 4/4, we use accordingly fewer bins (e.g., 48 bins for a 3/4 metre).

4. In addition to these 'bar-level' patterns, compute the average of all bars that belong to a song to obtain a 'song-level' pattern. These will be beneficial for music with constant rhythm like Ballroom music.

5. Normalize each feature to have zero mean and unit standard deviation.

6. Divide bar- and song-level patterns into R clusters using the k-means method.

[2]We might know that audio frames k and m are at the first and second beat of a bar, but do not know which bar position exactly an audio frame l, $k < l < m$, has.

7. Using the cluster labels for each instance, learn the parameters of $P(\mathbf{y}_k|m_k, r_k)$, $p_r(r_{k-1}, r_k)$, $n_{min}(r_k)$ and $n_{max}(r_k)$, representing the rhythmic patterns, their interaction and the tempo range of a pattern respectively.

The resulting patterns constitute the initial model that is fed into the model refinement or learning process.

4.3.2 Model Refinement

The semantics of the hidden state sequence for a data sample determine its segmentation. We thus aim to increase the probability of obtaining the correct state sequence when decoding the given observation sequence. To this end, we apply Viterbi training (Rabiner et al., 1986), which adapts the model's parameters such that the probability of the decoded state sequence increases. To ensure that the decoded state sequence corresponds to the desired (correct) segmentation result, we apply methods for *Partially-Hidden Markov Models* (Ramasso, 2009), which allow us to narrow down possible decoding paths to a specified window around the annotated beat structure.

Viterbi Training

Viterbi training works similarly to the well-known Baum-Welch method, with minor but critical alterations. It was first described under the name *segmental k-means algorithm* by Rabiner et al. (1986). The learning process repeats the following two steps until convergence: 1) decoding using the Viterbi algorithm and 2) parameter re-estimation. In doing so, at each step it selects new model parameters $\bar{\lambda}$ such that

$$\bar{\lambda} = \arg\max_{\lambda} \{ P(\mathbf{x}^*_{1:K} \mid \mathbf{y}_{1:K}, \lambda) \}, \tag{4.5}$$

where λ are the current model's parameters, and $\mathbf{x}^*_{1:K}$ is the decoded state sequence given by the Viterbi algorithm as defined in Eq. 4.2.

After decoding, the algorithm re-estimates the parameters in a straightforward way: p_n and $p_r(r_{k-1}, r_k)$ are computed by counting the corresponding occurrences in the decoded paths; similarly, we find p_n, the parameter defining the probability of a tempo change in tempo, and $n_{min}(r_k)$ and $n_{max}(r_k)$, which define the possible tempo range for a given pattern. Rhythmic patterns are created using the same method as in the initialisation, with the respective r_k instead of the clustering results as labels. The re-estimated parameters maximise the probability of the decoded state sequence.

Compared to the Baum-Welch algorithm, Viterbi training has some favourable properties for our use case. First and foremost, it uses a different objective function, as Eq. 4.5 shows. Instead of maximising the likelihood of the data (maximum likelihood

estimation, MLE), it maximises the a-posteriori probability (MAP) of the most probable state sequence. MLE is more closely related to a generative scenario - improvements in segmentation quality come incidentally, but are not the target of the optimisation. MAP maximisation, however, is related to the very problem we address in this paper - we use it to increase the probability of obtaining a state sequence that corresponds to the correct segmentation. Rodríguez and Torres (2003) also show that under certain circumstances, Viterbi training can lead to better segmentation results. Allahverdyan and Galstyan (2011) state that Viterbi training converges faster, and results in sparser solutions.

Decoding in Partially-Hidden Markov Models

The standard formulations of the common inference algorithms for HMMs are based upon the assumption that only the observed variables are known. Parameter learning methods such as the Baum-Welch algorithm do not require – and cannot incorporate – knowledge about the hidden variables of a model. Ramasso (2009) modified these algorithms such that uncertain and imprecise labels, in our case beat annotations, can be utilised for inference and to support learning of HMMs. Here, we use *evidential Viterbi decoding* in the model refinement step of our training method. Due to space limitations, we can only briefly describe the difference from regular Viterbi decoding. For an in-depth description, we refer to Ramasso (2009).

At each new time step k, the Viterbi algorithm computes *Viterbi messages* $\delta_k(s)$, which represent the probability of the most probable state sequence that ends at state s. Based on our belief of the true bar position at this time, we can also define the *plausibility* $pl_k(s)$ of being at state s at time step k. As derived by Ramasso (2009), the Viterbi messages in the evidential Viterbi algorithm are

$$\delta'_k(s) = \delta_k(s) \cdot pl_k(s),$$

while the remaining steps of the algorithm are unmodified.

We define the plausibilities based on the ground truth annotations. This results in uniform plausibilities at time steps between beats, causing the algorithm to behave like the standard decoding method. The behaviour changes within a range around the beat annotations: We force the decoded path to hit the annotated beat within $\pm$ 17.5% of the current inter-annotation interval around it. To this end, we assign a plausibility of zero to states corresponding to positions outside of this range, while keeping it uniform for those within. The allowed range corresponds to the tolerance window of the CMLt evaluation metric (see Section 4.4.2).

Thus we can alleviate the impact of imprecise annotations while forcing the algorithm to follow the correct segmentation path within the allowed tolerance window.

4.4 Experiments

In this section, we describe the experimental set-up to show the effect of the proposed parameter learning scheme on beat and downbeat tracking accuracy. The experimental results are obtained using a 10-fold cross-validation, where each fold was designed to have the same distribution of dance styles or metre. We stoped the refinement when the beat tracking F-measure on the validation set began to decrease.

We used a multiple comparison using one-way anova with a tukey-kramer correction (as provided by the *multcompare* tool of MATLAB (Hochberg and Tamhane, 1987)) to determine whether the difference between the group means are statistically significant with an alpha level of 0.05. All results which are statistically equivalent to the best result are printed in bold. Because of the rhythmic continuity, we used 'song-level' patterns for the Ballroom dataset, and 'bar-level' patterns for the Hainsworth dataset, in the training stage. The datasets are described in the following.

4.4.1 Datasets

We evaluated the proposed learning scheme on two datasets which provide beat and downbeat annotations.

The **Ballroom** dataset comprises ballroom dance music and is expected to consist of clear and constant rhythmic patterns, which makes it suitable for pattern extraction and modeling tasks. It was originally assembled for the tempo induction contest organised during the International Conference on Music Information Retrieval (ISMIR 2004) (Gouyon et al., 2006). The dataset consists of 685 unique excerpts of music, each 30 seconds long. [3]

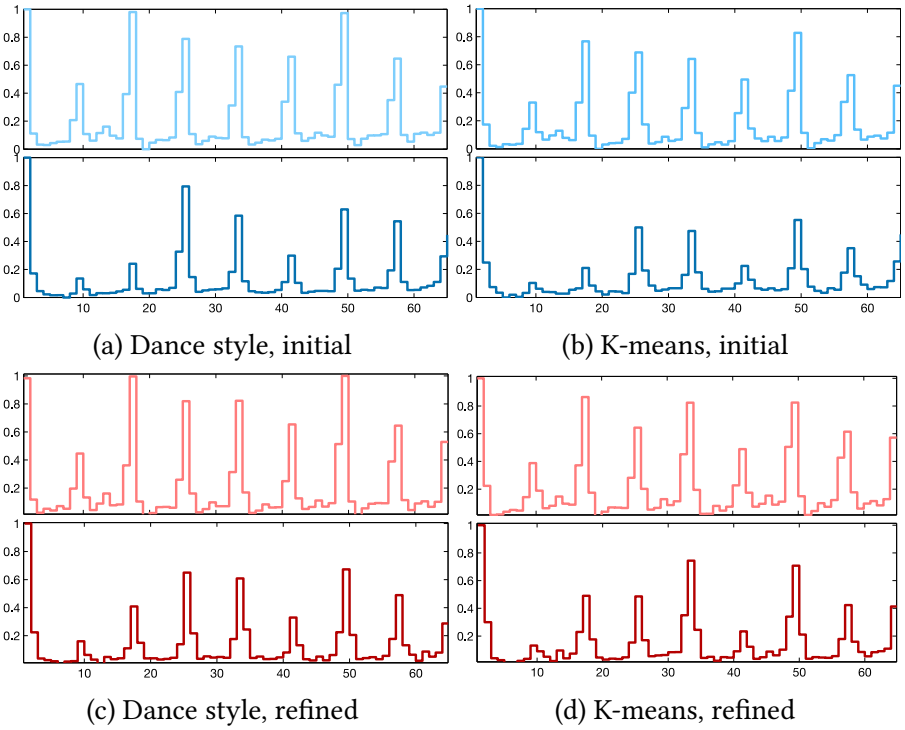
In order to investigate whether the proposed learning scheme also works for music with less constant rhythmic patterns, we chose to use the **Hainsworth** dataset, which has also been used by several other authors, e.g. (Hainsworth and Macleod, 2004; Peeters and Papadopoulos, 2011). It consists of 222 audio files of various genres, each between 30 and 60 seconds long.

4.4.2 Evaluation metrics

A variety of measures for quantifying the performance of beat tracking systems exist (Davies et al., 2009). We give results for the following four metrics:

FM is the generic F-measure often used in the field of information retrieval applied to the beat detection task. A detected beat is considered correct if it falls within a $\pm$ 70ms window around an annotated one.

[3] Annotations available at https://github.com/CPJKU/BallroomAnnotations (v1.1)



(a) Dance style, initial (b) K-means, initial

(c) Dance style, refined (d) K-means, refined

Figure 4.1: Two learned rhythmic patterns, before (Figs. a, b) and after refinement (Figs. c, d). The x-axis defines the position within a bar in a 64^{th} note grid, the y-axis represents the normalised mean onset values for a position. Each pattern consists of two frequency bands: the lower, darker plots represent the low band, the lighter, upper plots the high band.

CMLt is a continuity-based metric that describes the percentage of beats that belong to a continuous group. For a beat to belong to a continuous group, the previous and the current beat have to be within the tolerance window of $\pm$ 17.5% of the current inter-annotation interval around the annotations. Only beats at the correct metrical level are taken into account.

AMLt is computed the same way as CMLt, but beats are also allowed to occur at half or double the correct tempo, as well as on the offbeat.

DBFM is the downbeat F-measure, computed the same way as the beat F-measure, but using downbeats instead of beats.

4.5 Results and discussion

In Tables 4.1 and 4.2 we show results for the following systems: For the Ballroom dataset, **PS8_dancestyles**, the model proposed by Krebs et al. (2013), where dance

System	FM	CMLt	AMLt	DBFM
Krebs et al. (2013)	85.7	75.1	**88.3**	**69.7**
Krebs et al. (2013) (refined)	**87.5**	**78.5**	**89.1**	**69.9**
PS8_kmeans_b	85.3	74.0	**87.4**	63.3
PS8_kmeans_b (refined)	**87.3**	**78.2**	**88.4**	**70.7**
Peeters and Papadopoulos (2011)	76.3	56.7	84.1	42.1
Böck and Schedl (2011) (retrained)	**89.4**	**80.7**	84.4	-

Table 4.1: Beat and downbeat tracking accuracy on the *Ballroom* dataset. Bold printed results are statistically equivalent to the best performing one.

style labels are used to learn the rhythmic patterns[4]; **PS8_kmeans_b**, which uses our proposed pattern learning method with K=8 patterns trained on the Ballroom dataset; For the Hainsworth dataset, **PS2_metre**, which uses K=2 patterns, where each corresponds to a time signature present in the dataset (3/4 and 4/4); **PS5_kmeans** and **PS10_kmeans**, with 5 and 10 patterns respectively, all trained only on the Hainsworth dataset. Systems that use the proposed refinement stage (Section 4.3.2) are marked by the word **refined** in brackets. Additionally, we show results for the systems proposed by Peeters and Papadopoulos (2011) and Böck and Schedl (2011) as reference. We obtained beat detections for the reference systems directly from the respective authors. According to Böck and Schedl (2011), their system has been retrained since its first publication and uses now both Ballroom and Hainsworth dataset (among others) for training.

Ballrom dataset As can be seen from Table 4.1, the proposed PS8_kmeans_b method performs equally well as the PS8_dancestyles model in all metrics except DBFM. This is an interesting finding and suggests that with the proposed training method we no longer require dance-style labels for training the model. In addition, the refinement based on Viterbi training led to an improvement of the accuracies in all cases, even the ones that used dance-style labels for initialisation. However, the improvement was only found to be statistically significant for the CMLt and DBFM metric. A visual inspection of the learned rhythmic patterns reveals that the unsupervised training process yields patterns that are similar to those using the dance-style labels for initialisation (compare left and right columns of Fig. 4.1, for example).

In comparison to the reference systems, the proposed system performs comparable to the system of Böck and Schedl (2011) in the metrics FM and CMLt, while outperforming both Böck's and Peeter's system in the AMLt metric. It also achieves higher

[4]Results vary from those published earlier because of a different training strategy (10-fold cross-validation vs. leave-one-out).

DBFM scores than Peeter's system. However, the comparison with Peeters and Papadopoulos (2011) is somewhat unfair as their system did not have access to ballroom music for training.

System	FM	CMLt	AMLt	DBFM
PS2_metre	73.2	55.6	**79.4**	**40.7**
PS2_metre (refined)	73.5	57.3	**80.1**	**40.6**
PS5_kmeans	75.4	61.3	**82.6**	**44.6**
PS5_kmeans (refined)	75.1	62.2	**82.6**	**45.5**
PS10_kmeans	**76.4**	64.3	**83.9**	**43.5**
PS10_kmeans (refined)	**76.0**	61.9	**82.8**	**43.5**
Peeters and Papadopoulos (2011)	72.0	60.6	**84.3**	**43.8**
Böck and Schedl (2011)	**85.4**	**75.5**	**83.2**	-

Table 4.2: Beat and downbeat tracking accuracy on the *Hainsworth* dataset. Bold printed results are statistically equivalent to the best performing one.

Hainsworth dataset Table 4.2 shows the results on the Hainsworth dataset. Comparing the models with 2, 5, and 10 rhythmic patterns, we can see that the more patterns we use, the higher the accuracy gets, at least for the beat tracking metrics. Apparently, the k-means clustering is able to find musically relevant rhythmic patterns also in this dataset. Interestingly, the proposed refinement stage does not increase the average performance of our model on the Hainsworth dataset.

Compared to the other reference systems, we find that Böck's system outperforms all other systems in the metrics FM and CMLt, although the difference to the PS10_kmeans system is only statistically significant for the CMLt metric. One reason for the superior performance of Böck's system might be that it was trained on a larger dataset than the PS8_kmeans model, which was trained *only* on the Hainsworth dataset. Future work should therefore concentrate on learning rhythmic patterns on a larger dataset. Peeter's system scores best in the AMLt metric, but the difference to the other systems is not statistically significant. Also for the DBFM metric we do not find any significant differences among the systems.

4.6 Conclusion

In this paper, we have proposed a method of learning the parameters of the rhythm analysis model described by Krebs et al. (2013). The new method learns suitable patterns in an unsupervised way, given only downbeat and beat locations. We have shown

that the proposed unsupervised method achieves results comparable to those of the supervised model on the Ballroom dataset. On the Hainsworth dataset, the proposed method performs comparable to the reference systems, but is outperformed by Böck and Schedl (2011) in the CMLt metric.

While the clustering stage of our methods performs as expected, it is still unclear under which conditions the proposed refinement increases the performance of a model. While we found a significant improvement in the CMLt and the DBFM metrics when using Viterbi refinement on the Ballroom dataset, results are less clear-cut on the Hainsworth dataset.

In future work we will therefore concentrate on learning rhythmic patterns from a larger dataset and will investigate how the optimal number of rhythmic patterns can be found for a given dataset.

Chapter 5

Inferring Metrical Structure in Music Using Particle Filters

Published This is the author's version of an article that has been published in IEEE/-ACM Transactions on Audio, Speech and Language Processing (Krebs et al., 2015b). Changes were made to this version by the publisher prior to publication. The final version of record is available at http://dx.doi.org/10.1109/TASLP.2015.2409737.

Authors Florian Krebs, Andre Holzapfel, Ali Taylan Cemgil, and Gerhard Widmer.

Personal contributions The idea of applying a mixture particle filter to the meter tracking problem arose in discussions between me, Andre and Taylan. I did the implementations and ran the experiments.

Abstract In this work, we propose a new state-of-the-art particle filter (PF) system to infer the metrical structure of musical audio signals. The new inference method is designed to overcome the problem of PFs in multi-modal probability distributions, which arise due to tempo and phase ambiguities in musical rhythm representations. We compare the new method with a hidden Markov model (HMM) system and several other PF schemes in terms of performance, speed and scalability on several audio datasets. We demonstrate that using the proposed system the computational complexity can be reduced drastically in comparison to the HMM while maintaining the same order of beat tracking accuracy. Therefore, for the first time, the proposed system allows fast

meter inference in a high-dimensional state space, spanned by the three components of tempo, type of rhythm, and position in a metric cycle.

5.1 Introduction

Automatic inference of metrical structure from musical audio has been an active research topic over the last few decades. Especially, estimating the perceptually most salient pulsation in musical meter, i.e., the beat, is one of the aspects that has attracted a significant amount of research work (see Müller et al. (2011) for an overview). While also interesting in its own right, the automatic determination of the metrical grid from an audio recording is a fundamental ingredient for other, high-level Music Information Retrieval (MIR) tasks, such as music genre recognition (Dixon et al., 2004), chord estimation (Mauch and Dixon, 2010), and music transcription (Cemgil et al., 2006).

Over the last decade, probabilistic state-space models have become a popular framework to tackle the metrical inference problem (e.g., Klapuri et al. (2006); Peeters and Papadopoulos (2011); Degara et al. (2011)). In these models it is attempted to infer a set of hidden states (such as beat times, tempo, meter) from a set of observed states (such as estimated note onset times and/or other audio features). Bayesian methods allow to easily represent the ambiguity that is inherent to musical meter and offer a consistent framework to combine multiple sources of information (e.g., preferred tempi, knowledge about note onset locations). Nevertheless, exact inference in these models is only feasible in a few simple cases, e.g., in discrete state spaces using hidden Markov models (HMMs) (Rabiner, 1989) or for linear Gaussian conditional distributions (Roweis and Ghahramani, 1999).

To overcome these limitations, approximative methods such as particle filters (PF) (Doucet and Johansen, 2009) have been proposed. Particle filters are a highly efficient method that can be applied to arbitrary, high-dimensional, non-linear, non-Gaussian state-spaces. Consequently, the use of particle methods makes it possible to use more complex models, which has also been exploited for the beat tracking task: Cemgil and Kappen (2003); Hainsworth and Macleod (2004); Lang and de Freitas (2004) model the beat times and tempo jointly, taking into account their mutual dependency. Furthermore, Whiteley et al. (2007) introduced a rhythmic pattern state, which allows modelling various meters and rhythmic styles explicitly. In any case, the great flexibility of the PFs comes at a prize: Simple PF schemes are known to perform poorly with multi-modal probability distributions, which arise from ambiguity in the hidden state-sequence that generated the data (Vermaak et al., 2003). Therefore, much work has been devoted to resolving this problem, and several extensions to the simple PF have been proposed (Pitt and Shephard, 1999; Vermaak et al., 2003; Doucet and Johansen, 2009). However, although multi-modal distributions appear frequently in analysis of musical rhythm due to inherent tempo and phase ambiguities, dealing with this multi-

modality with PFs in a robust way has never been addressed in the meter tracking literature.

In this work, we aim at setting a new state of the art in PF based meter inference systems. We propose a new particle filter based inference method to overcome the problem of tracking a multi-modal distribution by combining the auxiliary (APF) (Pitt and Shephard, 1999) and the mixture particle filter (MPF) (Vermaak et al., 2003), and compare the resulting auxiliary MPF (AMPF) with the HMM system proposed by Krebs et al. (2013) and several other PF schemes in terms of performance, speed and scalability on several audio datasets. We demonstrate that using the AMPF the computational complexity can be reduced drastically compared to the HMM while maintaining the same order of beat tracking accuracy. Apart from beat tracking, our system is capable of determining downbeat times, time signature and type of rhythm of an audio piece using one integrated approach. Thus, for the first time, the proposed AMPF permits fast inference in the high-dimensional state-space spanned by variables describing tempo, position within a metric cycle, and type of the metric cycle.

The structure of the underlying model presented in this paper can be easily adapted to the music style under investigation, and parameters of the model can be learned offline from rhythmic patterns encountered in a representative music corpus. This will be described in more detail in Section 5.3.4. In most existing approaches, parameters and structure of the beat tracking systems are tailored by experts, with the goal to cope well with the demands of specific annotated evaluation datasets. This is problematic under at least two aspects. First, the manual adaption to a specific style is time-consuming and hence is not a viable solution for covering a wider variety of styles. Second, such systems are not suited for later adaption by users, and therefore incorporate the risk to include a bias (Bozdag, 2013) that systematically discriminates musical styles not considered during development. Therefore, our approach represents an important step towards flexible representations of musical concepts that can be adapted by incorporating available knowledge of musical style in a straight-forward way, as demanded by Serra et al. (2013).

In the following sections, we will introduce basic notions of the metrical structure of music, and explain the structure of our dynamic Bayesian network by defining the hidden variables, the transition model and the observation model. In Section 5.4 we describe different ways to perform inference in this model. First, the HMM (Section 5.4.1) represents an accurate inference approach with high computational cost. To reduce this cost, we describe several inference schemes based on PFs in Section 5.4.2. Section 5.5 describes datasets and evaluation metrics and the default settings for all system parameters. Experimental results are presented in Section 5.6, with a focus on the comparison of the HMM inference with the PF schemes. Section 5.7 provides the reader with a detailed summary and discussion of the experimental results. The final Section 5.8 concludes the paper and outlines future directions of research motivated

by our results.

5.2 Metrical structure of music

Meter as defined by Kolinski (1973) is organized pulsation functioning as a framework for rhythmic design. Especially in the context of Eurogenetic music, this pulsation is considered to be stratified into a hierarchy, with the period of pulsation increasing towards higher layers. The pulsation of the *beat* is situated in a layer in the middle of this hierarchy, and represents the rate that listeners choose to synchronize their body movements to the music. Beats are grouped into segments with a constant number of beats (*bars*), defined by the *time signature*. The first beat of each bar is denoted the *downbeat* (see Fig. 5.1 for an illustration). Listeners differ in their perception of meter, caused by individual or cultural factors (Stobart and Cross, 2000). For instance, the perceived beat can be related to different metrical layers, which results in perceived tempi that are typically related by a factor of two. This variability must be considered both in the implementation of inference systems and in their evaluation.

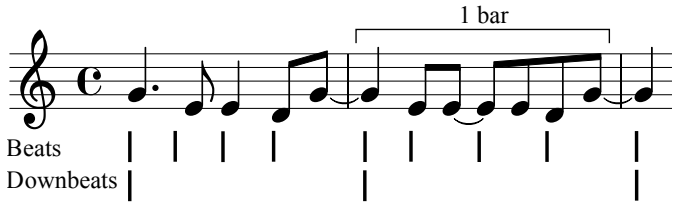


Figure 5.1: Illustration of beats and downbeats in a musical score

Computational inference of meter can either be approached in an *on-line* or *off-line* fashion. On-line tracking requires inference at the same moment of observing the musical sound, without the possibility of looking into the future. On the other hand, off-line processing assumes that a recording of the whole piece is given, and for the meter inference at a specific moment in the recording the future can be taken into account. In this paper, we evaluate our system in off-line mode but the proposed methodology can be applied to on-line scenarios as well.

5.3 Model structure

In this section, we formulate the metrical structure analysis problem using a Bayesian model. We assume that a time series of *observed* data $\mathbf{y}_{1:K} = \{\mathbf{y}_1, ..., \mathbf{y}_K\}$ (the audio signal as a sequence of audio features) is generated by a set of unknown, *hidden* variables $\mathbf{x}_{1:K} = \{\mathbf{x}_1, ..., \mathbf{x}_K\}$ (the parameters describing tempo and meter throughout the

progression of a piece), where K is the length of an audio excerpt in analysis frames. In a *dynamic Bayesian network* (DBN) (Murphy, 2002), the joint probability distribution of hidden and observed variables $P(\mathbf{y}_{1:K}, \mathbf{x}_{1:K})$ then factorizes as

$$P(\mathbf{y}_{1:K}, \mathbf{x}_{1:K}) = P(\mathbf{x}_1) \prod_{k=2}^{K} P(\mathbf{x}_k|\mathbf{x}_{k-1}) P(\mathbf{y}_k|\mathbf{x}_k), \tag{5.1}$$

where $P(\mathbf{x}_1)$ is the *initial state distribution*, $P(\mathbf{x}_k|\mathbf{x}_{k-1})$ is the *transition model*, and $P(\mathbf{y}_k|\mathbf{x}_k)$ is the *observation model*. We will describe these three terms in more detail in Sections 5.3.2 to 5.3.4, after presenting the internal structure of the hidden variables in $\mathbf{x}$ in subsection 5.3.1.

5.3.1 Hidden variables

The model described in this paper closely follows the bar pointer model proposed by Whiteley et al. (2006). In this model, the observation at each time step k is a short audio frame, and the hidden variables describe the state of a hypothetical bar pointer $\mathbf{x}_k = [\phi_k \; \dot{\phi}_k \; r_k]$ corresponding to the k-th audio frame; the variable ϕ_k is the current location in a bar, $\dot{\phi}_k$ is the instantaneous tempo (denoting the rate at which the bar pointer traverses a bar), and r_k is a rhythmic pattern indicator that can be used to differentiate between time signatures or between rhythmic styles of identical time signature. Below, we make these definitions more precise:

Bar position

We define the bar position $\phi \in [0, \theta_{max})$, where θ_{max} is the length of a bar related to the longest considered metric cycle in the data. For example, if the time signatures of the considered meters are $9/8$, $4/4$, and $3/4$, we set $\theta_{max} = 9/8$.

Tempo

We define tempo $\dot{\phi}_k \in [\dot{\phi}_{min}(r_k), \dot{\phi}_{max}(r_k)]$ in terms of beats per minute (bpm); The tempo limits are assumed to depend on the rhythmic pattern state r_k and are learned from data (e.g., for the rhythmic pattern variable r_k assigned to a Tango pattern we may find $\dot{\phi}_k \in [118bpm, 136bpm]$).

Rhythmic pattern

The rhythmic pattern variable $r_k \in [1...R]$ is an indicator, which selects one of the R underlying observation models. Each observation model is associated with a time signature $\theta(r)$ (e.g., $\theta(r) = 3/4$) and a specific rhythmic structure that is learned from data. Note that there can be several rhythmic patterns sharing the same time signature.

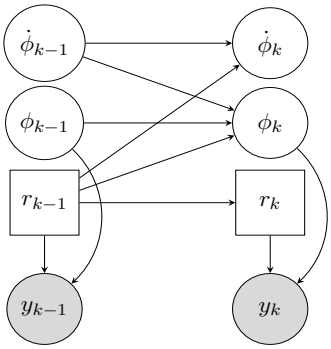


Figure 5.2: Dynamic Bayesian network; circles denote continuous variables and rectangles discrete variables. The gray nodes are observed, and the white nodes represent the hidden variables.

An example of two such learned patterns is given in Fig. 5.4, along with a detailed description of the observation models in Section 5.3.4.

The conditional independence relations between these variables are shown in Fig. 5.2. The hidden state sequence, inferred from an audio piece, can finally be translated into a sequence of *time signature(s)* $\theta(r_k)$, *downbeat times* (time frames that correspond to $\phi_k = 0$), and *beat times* (time frames that correspond to $\phi_k = i \cdot (denom(\theta(r_k)))^{-1}, i = 1, 2, ..., num(\theta(r_k))$, where $denom$ and num are the denominator and numerator, respectively).

5.3.2 Initial state distribution

Using the initial state distribution $P(\mathbf{x}_1)$, *a priori* knowledge regarding rhythmic aspects can be introduced into the system. The case that certain rhythmic patterns are encountered more frequently can be modeled, or certain tempo values can be preferred using a weighting function (Klapuri et al., 2006). For the experiments in this paper, we have simply assumed uniformly distributed bar position, tempo, and rhythmic patterns states within the learned tempo ranges $[\dot{\phi}_{min}(r_k), \dot{\phi}_{max}(r_k)]$.

5.3.3 Transition model

Due to the conditional independence relations shown in Fig. 5.2, the transition model factorizes as

$$\begin{aligned} P(\mathbf{x}_k|\mathbf{x}_{k-1}) = \quad & P(\phi_k|\phi_{k-1}, \dot{\phi}_{k-1}, r_{k-1}) \times \\ & \times P(\dot{\phi}_k|\dot{\phi}_{k-1}, r_{k-1}) \times P(r_k|r_{k-1}) \end{aligned} \tag{5.2}$$

where the three factors are defined by Equations 5.3-5.5:

$$P(\phi_k \mid \phi_{k-1}, \dot{\phi}_{k-1}, r_{k-1}) = \mathbb{1}_x, \tag{5.3}$$

where $\mathbb{1}_x$ is an indicator function that equals one if $\phi_k = (\phi_{k-1} + \dot{\phi}_{k-1} \cdot \Delta \cdot (60 \cdot denom(\theta(r_{k-1})))^{-1}) \bmod \theta(r_{k-1})$, and zero otherwise, with $\Delta = 0.02s$ the audio frame length used in this paper. This means that the bar position at frame k is obtained by increasing the bar position of the previous frame by a term that depends on the tempo of the previous frame. The tempo transition from one frame to the next is assumed to follow a normal distribution and is given by

$$P(\dot{\phi}_k|\dot{\phi}_{k-1}, r_{k-1}) \propto \mathcal{N}(\dot{\phi}_{k-1}, \sigma^2_{\dot{\phi}}) \times \mathbb{1}_y, \tag{5.4}$$

where $\sigma_{\dot{\phi}}$ is the standard deviation of the tempo transition model and $\mathbb{1}_y$ is an indicator function that equals one if $\dot{\phi}_{min}(r_{k-1}) \leq \dot{\phi}_k \leq \dot{\phi}_{max}(r_{k-1})$, and zero otherwise.

$$P(r_k|r_{k-1}) = \mathbb{1}_z, \tag{5.5}$$

where $\mathbb{1}_z$ is an indicator function that equals one if $r_{k+1} = r_k$, and zero otherwise. This means that we assume a musical piece to have a characteristic rhythmic pattern that remains constant throughout the song. This rather strict assumption can be relaxed by defining a rhythmic pattern transition probability that is non-zero at bar boundaries (Whiteley et al., 2006).

5.3.4 Observation model

As proposed by Krebs et al. (2013), we use an onset feature as observed variable $\mathbf{y}_k$, which we assume to be independent of the current tempo $\dot{\phi}_k$. Therefore, the observation model $P(\mathbf{y}_k|\mathbf{x}_k)$ reduces to $P(\mathbf{y}_k|\phi_k, r_k)$, which means that the probability of observing a certain feature value at a given time-point depends only on the rhythmic style and the position in a bar. In order to obtain the parameters of $P(\mathbf{y}_k|\phi_k, r_k)$, a collection of beat- and downbeat annotated audio samples is needed (see Section 5.5.3 for details on the training set we use in this paper).

Observation features

As observation feature, we use a variant of the *LogFiltSpecFlux* onset feature, which performed well in recent comparisons of onset detection functions (Böck et al., 2012b) and is summarized in Fig. 5.3. Assuming that the bass instruments play an important role in defining rhythmic patterns, we compute the sum over frequency bands separately in low frequencies (below 250 Hz) and high frequencies (above 250 Hz). Finally, we subtract the moving average computed over a window of one second and normalize the features of each excerpt to zero mean and unit variance, again separately for the low and high frequencies. The resulting onset feature $\mathbf{y}_k$ has therefore two dimensions.

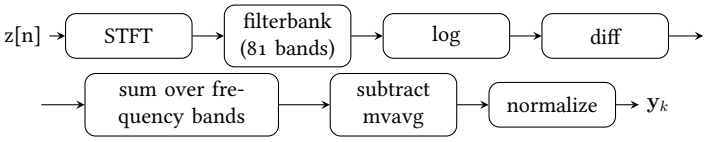


Figure 5.3: Computing the onset feature $\mathbf{y}_k$ from the audio signal $z[n]$

Likelihood function

The parameters of the observation model are obtained in an off-line (supervised) rhythm pattern learning process. In order to get a modest (and computationally feasible) number of probability distributions to represent $P(\mathbf{y}_k|\phi_k, r_k)$, we discretize the bar position Φ into 64th note cells. In order to learn the parameters of the likelihood function, a set of training pieces must be available that is annotated at the beat and bar layers of the metrical hierarchy. Using these annotations, we can assign each feature vector $\mathbf{y}_k$ to the corresponding rhythmic pattern and bar position within the 64th note grid. Then, for each bar position within this 64th grid and each rhythmic pattern, we compute the maximum likelihood estimates of the parameters in a Gaussian mixture model (GMM). As suggested by Krebs et al. (2013), we use $I = 2$ mixture components in this work. Hence, the observation probability is modelled by

$$P(\mathbf{y}|\phi, r) = \sum_{i=1}^{I} w_{\phi,r,i} \cdot \mathcal{N}(\mathbf{y}; \mu_{\phi,r,i}, \Sigma_{\phi,r,i}), \tag{5.6}$$

where $\boldsymbol{\mu}_{\phi,r,i}$ is the (2-dimensional) mean vector, $\boldsymbol{\Sigma}_{\phi,r,i}$ is the (2×2) covariance matrix, and $w_{\phi,r,i}$ is the mixture weight of component i of the GMM. Since, in learning the likelihood function $P(\mathbf{y}|\phi, r)$, a GMM is fitted to the audio features for every rhythmic pattern r and each 64th note bar position cell, the resulting GMMs can be interpreted directly as representations of the rhythmic patterns in the domain of the observation variables. Fig. 5.4 shows the mean values of the features per frequency band and bar position for the GMMs corresponding to the rhythmic patterns of a Waltz (3/4) and a Tango dance (4/4), respectively. For illustration purposes we only display the mean value of the *features* instead of the mean values per mixture component.

5.4 Inference methods

Our goal is to find the hidden state sequence that maximizes the (posterior) probability of the hidden states given the observations $P(\mathbf{x}_{1:K}|\mathbf{y}_{1:K})$. If we discretize the continuous tempo and bar pointer variables (see Section 5.3.1), we can in principle perform an exact inference using an HMM (Section 5.4.1).

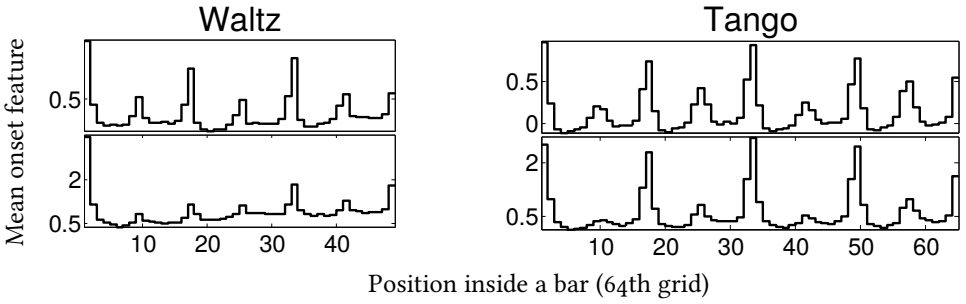


Figure 5.4: Illustration of two learned rhythmic patterns. Two frequency bands are shown (Low/High from bottom to top).

However, in order to avoid the high computational complexity of the HMM inference illustrated in Section 5.4.1, we describe approaches for inference using PFs in Section 5.4.2. We begin with PF approaches widely discussed in literature, and then present novel approaches capable of tracking the metrical structure in the highly multimodal posterior distributions typical for musical rhythm.

5.4.1 Hidden Markov Model (HMM)

Inference in the model discussed in Section 5.3 can be performed using an HMM by dividing the state space into discrete cells and using Viterbi decoding (Rabiner, 1989) to obtain the *maximum a posteriori* (MAP) sequence of states. In Fig. 5.5 we show a realization of a bar position/tempo trajectory and a possible discretization.

In this work, we use the discretization proposed by Whiteley et al. (2006), which we further explain in this section. By replacing the continuous variables ϕ and $\dot{\phi}$ by their discretized counterparts $m \in \{1, ..., M\}$ and $n \in \{1, ..., N\}$ respectively, Equations 5.2, 5.3 and 5.5 remain valid. We only define a new tempo transition probability as:
If $n_{min}(r_k) \leq n_k \leq n_{max}(r_k)$,

$$P(n_k|n_{k-1}) = \begin{cases} 1 - p_n, & n_k = n_{k-1}; \\ \frac{p_n}{2}, & n_k = n_{k-1} + 1; \\ \frac{p_n}{2}, & n_k = n_{k-1} - 1, \end{cases} \tag{5.7}$$

otherwise $P(n_k|n_{k-1}) = 0$.

Here, p_n is the probability of a tempo change and $n_{min}(r_k)$ and $n_{max}(r_k)$ are the discrete tempo limits that correspond to $\dot{\phi}_{min}$ and $\dot{\phi}_{max}$.

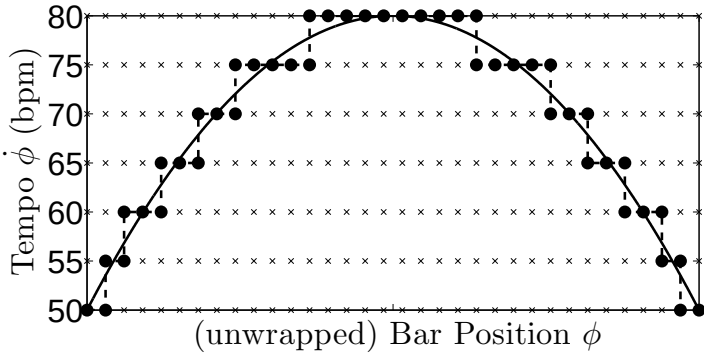


Figure 5.5: Illustration of discretization of the tempo and bar pointer variables. The continuous line depicts a tempo trajectory that might be encountered in the expressive timing throughout a musical phrase, and the the dashed line demarks the trajectory through the discretized states.

The HMM is most accurate when the discretization grid is dense, but this can become computationally prohibitive. For instance, discretizing the bar position ϕ of a 4/4 bar into $M = 1200$ points (300 per quarter note) and the tempo $\dot{\phi}$ into 23 points, results in a state-space of $S = M \cdot N = 27\,968$. Adding more rhythmic pattern states increases the dimensionality to $M \cdot N \cdot R$, which quickly surpasses the computational and memory limits of a current personal computer. This problem can be overcome by applying approximate inference schemes instead of the exact HMM inference, and we will present such schemes in the following parts of this section.

5.4.2 Particle filter (PF)

Even though the exact computation of the posterior $P(\mathbf{x}_{1:K}|\mathbf{y}_{1:K})$ in the continuous parameter space is intractable, it can nevertheless be evaluated point-wise. This fact is exploited in the PF where the posterior is approximated by a weighted sum of points (i.e., particles) in the state space as

$$P(\mathbf{x}_{1:K}|\mathbf{y}_{1:K}) \approx \sum_{i=1}^{N_s} w_K^{(i)} \delta(\mathbf{x}_{1:K} - \mathbf{x}_{1:K}^{(i)}). \tag{5.8}$$

Here, $\{\mathbf{x}_{1:K}^{(i)}, i = 1, ..., N_s\}$ is a set of points with associated weights $\{w_K^{(i)}, i = 1, ..., N_s\}$ and $\mathbf{x}_{1:K}$ is the set of all states until time frame K, while $\delta(x)$ denotes the Dirac Delta function

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x \neq 0. \end{cases} \tag{5.9}$$

In order to approximate $P(\mathbf{x}_{1:K}|\mathbf{y}_{1:K})$ we need a strategy to draw samples $\mathbf{x}_k^{(i)}$ and compute appropriate weights $w_k^{(i)}$ recursively for each time k. A simple algorithm to do that for sequential data is *sequential importance sampling* (SIS) (Doucet and Johansen, 2009). In SIS, we sample from a *proposal* distribution $Q(\mathbf{x}_{1:K}|\mathbf{y}_{1:K})$ which should be as similar as possible to the true (*target*) distribution $P(\mathbf{x}_{1:K}|\mathbf{y}_{1:K})$. To correct for the fact that we sampled from the proposal instead of the target distribution, we assign an *importance weight* $w_K^{(i)}$ to each particle, which is computed by

$$w_K^{(i)} = \frac{P(\mathbf{x}_{1:K}|\mathbf{y}_{1:K})}{Q(\mathbf{x}_{1:K}|\mathbf{y}_{1:K})}. \tag{5.10}$$

If a suitable proposal density is chosen, these weights can be computed recursively by

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{P(\mathbf{y}_k|\mathbf{x}_k^{(i)})P(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})}{Q(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}, \mathbf{y}_k)}. \tag{5.11}$$

In this work, we chose to sample from the transition probability $Q(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}, \mathbf{y}_k) = P(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})$, which reduces Eq. 5.11 to

$$w_k^{(i)} \propto w_{k-1}^{(i)} P(\mathbf{y}_k|\mathbf{x}_k^{(i)}). \tag{5.12}$$

Then, the SIS algorithm derives samples and weights for time k by first drawing from the proposal, in our case $P(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})$, and then assigning weights according to (5.12).

Once the particle trajectories $\{\mathbf{x}_{1:K}\}$ have been determined, we select the particle trajectory $\mathbf{x}_{1:K}^{(i)}$ with the highest weight $w_K^{(i)}$ as the MAP state sequence. We have not attempted to improve the obtained state sequence by particle Viterbi decoding (Godsill et al., 2001) or particle smoothing (Doucet and Johansen, 2009) and leave this for future work.

Several extensions to the SIS filter have been proposed over the years (Doucet and Johansen, 2009). In the following, we will describe those approaches that we will evaluate for their applicability to metrical inference.

The sequential importance sampling/resampling (SISR) filter

The most challenging problem in particle filtering is to cope with the so called *degeneracy problem* (Doucet and Johansen, 2009): After some time, most of the particles have a weight close to zero, and thus represent very unlikely regions of the state space. This is in contrast to the ideal case with a perfect match between the proposal and target distribution, where the weights are uniformly distributed and thus have a low variance. In order to reduce the variance of the particles, it has been recommended to use *resampling* or *rejuvenation* steps, in order to replace particles with low weights

by particles with a higher weight. This is usually done by selecting particles with a probability that is proportional to their weights. Several schemes have been proposed in the literature, for a comparison see Douc et al. (2005). The most common resampling method is *systematic resampling* (Kitagawa, 1996) and has also been used in this paper. As recommended by Doucet and Johansen (2009), we do not resample at each iteration (*bootstrap filter*), but only perform resampling when the effective sample size

$$N_{ESS} = (\sum_{i=1}^{N_s} (w_k^{(i)})^2)^{-1} \tag{5.13}$$

is below a threshold of $\rho \cdot N_s$. The value of ρ will be determined together with other system parameters in Section 5.5.3.

Algorithm 1 Outline of the sequential importance sampling/resampling (SISR) filter, $\xi_k^{(i)}$ denotes $(\phi_k^{(i)}, r_k^{(i)})$

for $i = 1$ to N_s **do**
  Sample $(\phi_0^{(i)}, \dot{\phi}_0^{(i)}, r_0^{(i)}) \sim P(\phi_0)P(\dot{\phi}_0)P(r_0)$
  Set $w_0^{(i)} = 1/N_s$
end for
for $k = 1$ to K **do**
  for $i = 1$ to N_s **do** ▷ Proposal and weight computation
    Sample $\xi_k^{(i)} \sim P(\xi_k^{(i)}|\xi_{k-1}^{(i)})$
    $\tilde{w}_k^{(i)} = w_{k-1}^{(i)} \times P(\mathbf{y}_k|\xi_k^{(i)})$
  end for
  for $i = 1$ to N_s **do** ▷ Normalize weights
    $w_k^{(i)} = \frac{\tilde{w}_k^{(i)}}{\sum_{i=1}^{N_s} \tilde{w}_k^{(i)}}$
  end for
  Compute the effective sample size N_{ESS} (5.13)
  if $N_{ESS} \leq \rho \cdot N_s$ **then**
    for $i = 1$ to N_s **do**
      Resample $\{\xi_k^{(i)}, w_k^{(i)}\}$ to obtain $\{\xi_k'^{(i)}, 1/N_s\}$
    end for
    for $i = 1$ to N_s **do**
      $\xi_k^{(i)} = \xi_k'^{(i)}$
    end for
  end if
  Sample $\dot{\phi}_k^{(i)} \sim P(\dot{\phi}_k|\dot{\phi}_{k-1}^{(i)}, r_k^{(i)})$
end for

The essential difference between SIS and SISR is therefore the resampling. The SISR algorithm is outlined in Algorithm 1. Note that we sample the tempo state $\dot{\phi}$ at

the end of each iteration (after the resampling step). This is motivated by the general principle that any operation that does not influence the weights should take place after the resampling step in order to increase the diversity of the particles (Doucet and Johansen, 2009).

The Auxiliary Particle Filter (APF)

Although the introduction of resampling steps reduces the variance of the importance weights, the degeneracy problem is not yet solved. Resampling tends to lead to an extreme concentration of the particles at one particular mode of the posterior distribution, whereas the remaining distribution remains uncovered. This is particularly critical if the probability distribution has multiple modes as it is the case in our application.

One way to alleviate this problem is to compress the weights $\mathbf{w}_k = \{w_k^{(j)}, j = 1, ..., N_s\}$ by a monotonically increasing function g[1] before resampling. This increases the weights of particles at low probability regions and therefore makes it more probable that these particles survive the resampling. After resampling, the weights have to be uncompressed again in order to yield a valid probability distribution. This can be formulated in the terminology of the *auxiliary particle filter* (APF) (Johansen and Doucet, 2008), which can be summarized as follows:

- Compute the compressed weights by applying $g(\cdot)$, which causes a decrease of variance in the weights.
- Resample particles according to the compressed weights.
- Set each weight to the quotient of the uncompressed and the compressed weight.

The resampling procedure of the APF is sketched in Algorithm 2.

As an example, imagine a distribution of two particles $\{p_1, p_2\}$ with corresponding weights $\mathbf{w} = \{0.01, 0.99\}$. If we drew two samples from this distribution, it would be probable that p_1 vanishes from the particle set at the resampling step. However, if we modify the weights by the function $f(x) = x^{\frac{1}{4}}$ yielding $\hat{\mathbf{w}} \approx \{0.32, 1.00\}$, drawing p_1 becomes much more probable. Let us assume, we draw four samples from this (unnormalized) distribution $\hat{\mathbf{w}}$ and obtain the set $\{p_2, p_1, p_2, p_2\}$). In order to still represent the same distribution as before the sampling, we have to set the weight of each resampled particle to $x/f(x)$, which yields $\mathbf{w} \approx \{0.99, 0.03, 0.99, 0.99\}$.

Mixture particle filter (MPF)

As mentioned before, one major problem with applying particle filtering to the musical meter tracking problem is that the posterior distribution $P(\mathbf{x}_k|\mathbf{y}_{1:k})$ is highly multi-

[1] In this work, we restrict $g(w)$ to functions of the form w^β where $0 \leq \beta < 1$.

Algorithm 2 Outline of the resampling procedure of the auxiliary particle filter (APF), $\xi_k^{(i)}$ denotes $(\phi_k^{(i)}, r_k^{(i)})$

```
Compute the effective sample size N_ESS (5.13)
if N_ESS ≤ ρ · N_s then
    for i = 1 to N_s do
        ŵ_k^(i) = w_k^(i) g(ξ_k^(i))
    end for
    for i = 1 to N_s do
        Resample {ξ_k^(i), ŵ_k^(i)} to obtain {ξ'_k^(i), w_k^(i)/ŵ_k^(i)}
    end for
    for i = 1 to N_s do
        ξ_k^(i) = ξ'_k^(i)
    end for
end if
```

modal. A system that is able to cope with metrical ambiguities should maintain this multi-modality and track several hypotheses over a longer time.

Vermaak et al. (2003) proposed a system that tracks multiple football players in a video sequence using a *mixture PF*. Each particle is assigned to a cluster based on its location within the state space. Whenever it comes to resampling, particles interact only with particles of the same cluster (resampling of one cluster is performed independently of all other clusters). In this way, all modes that are covered by a cluster can be tracked successfully. In the following we describe an adaption of this method to the problem of finding the metrical structure in music.

At the beginning, we cluster the particles into C_0 clusters[2] by a run of the k-means clustering algorithm using a distance measure which takes into account the cyclic nature of the bar position ϕ. This means that a point at the beginning of a bar ($\phi \approx 0$) should be close to a point at the bar ending ($\phi \approx \theta$). Therefore, we represent the bar position as a complex phasor on the unit circle and compute the corresponding angle by

$$\alpha_k(\phi_k, r_k) = \frac{2\pi\phi_k}{\theta(r_k)}. \tag{5.14}$$

This angle can further be expressed by the periodic functions $\cos(\alpha)$ and $\sin(\alpha)$ using Euler's formula

$$e^{j\alpha} = \cos(\alpha) + j\sin(\alpha). \tag{5.15}$$

Using this transformation we define the distance measure for the k-means clustering

[2] In this paper, we start with 16 clusters per rhythmic pattern ($C_0 = 16 \cdot R$).

algorithm as

$$d(i,j) = \lambda_\phi \cdot [(\cos(\alpha^{(i)}) - \cos(\alpha^{(j)}))^2 + (\sin(\alpha^{(i)}) - \sin(\alpha^{(j)}))^2] + \lambda_{\dot{\phi}} \cdot (\dot{\phi}^{(i)} - \dot{\phi}^{(j)})^2 + \lambda_r \cdot (r^{(i)} - r^{(j)})^2,$$

where $[\alpha^{(i)}, \dot{\phi}^{(i)}, r^{(i)}]$ are the coordinates of the ith particle, and $\lambda_\phi, \lambda_{\dot{\phi}}, \lambda_r$ are coefficients that control the relative distance between the hidden variables. The assignment of particles to clusters is preserved until the next resampling step. Then, before resampling, the particles are reclustered in another run of k-means, using the old cluster assignments as initialization. Additionally, clusters are split, if the average distance from particle to cluster centroid is above a threshold τ_s and/or merged if the distance between two centroids is below a threshold τ_m. If the number of clusters exceeds a constant τ_c[3], the clusters with the lowest total weight are removed, assigning the affected particles to other clusters. These three operations are important to control the number of clusters, which should ideally represent the number of modes of the posterior distribution. In order to have a balanced number of particles per cluster, we draw the same number of particles for each cluster in the resampling step.

In contrast to the SISR and APF approaches, it does not make sense to determine the timing of the resampling based on the effective sample size N_{ESS}. If we computed the N_{ESS} on the whole particle set, mixture components with low total weights would constantly lead to a low N_{ESS} and therefore to more frequent resampling steps. Therefore, we chose to perform the resampling step with a fixed interval d.

Auxiliary Mixture particle filter (AMPF)

The AMPF combines the compression/decompression of the importance weights of the APF with the mixture tracking of the MPF.

5.5 Experimental Setup

5.5.1 Datasets

The performance evaluation of the PF algorithms and the HMM reference system requires annotated music recordings. We use three datasets for evaluation, which are frequently used in the MIR research community, and one for training:

The *SMC* dataset was presented by Holzapfel et al. (2012) and consists of 217 pieces that were considered to be difficult for automatic tempo tracking. Musical styles cover, e.g., French Chanson, classical orchestra music, and contemporary guitar music. For the *SMC* dataset, only beat annotations are available.

[3]Here, we used a maximum number of 50 clusters per rhythmic pattern ($\tau_c = 50 \cdot R$).

The largest dataset consists of 1360 songs and was compiled by Gouyon (2005) combining collections from various sources. We will refer to this dataset as the *1360-song* dataset throughout the text. It contains a wide range of musical styles of mainly Eurogenetic music, covering choral works of classical music as well as rock or jazz. The dataset is only annotated at the beat-level.

The third dataset used for evaluation is the *Ballroom* dataset, introduced by Gouyon et al. (2004) and annotated with beats and downbeats by Krebs et al. (2013). It contains 698 excerpts of ballroom dance music, along with dance style, beat, and downbeat annotations, which enables us to evaluate at both metrical levels. We removed 13 replicated excerpts (Sturm, 2014) to yield 685 unique ones. For some experiments, the dataset was split randomly into a test set (denoted *Ballroom_test*) of 204 songs and a training set (*Ballroom_train*) which consists of the remaining 481 songs, both having roughly the same genre distribution. The audio quality of this dataset is quite low (RealAudio format with high compression).

The fourth dataset which was used for training only is a collection of 97 files from the MIREX 2006 beat tracking contest, from Bello et al. (2005), and from Böck et al. (2012b). We denote this dataset as the *MBB* dataset. It covers the genres pop, rock and electronic music and is beat and downbeat annotated.

Due to the lack of downbeat annotations for the *SMC* and *1360-song* datasets, downbeat detection is only evaluated on the *Ballroom* dataset.

Train set	Test set
Ballroom+MBB	SMC
Ballroom+MBB	1360-song
Ballroom_train	Ballroom_test

Table 5.1: Train and corresponding test datasets.

5.5.2 Evaluation measures

We evaluate the HMM and PF inference schemes with respect to both *tracking accuracy* and *runtime*.

A variety of measures for beat and downbeat tracking performance is available (Davies et al., 2009). We chose four metrics that are characterized by a set of diverse properties and that are widely used in beat/downbeat tracking evaluation[4]. Furthermore, their choice enables for a direct comparison of the results on the *SMC* and the *1360-song* dataset with 16 reference systems (Holzapfel et al., 2012).

[4]We used the MATLAB evaluation code available at `http://code.soundsoftware.ac.uk/projects/beat-evaluation/` with standard settings to ensure reproducibility.

As our focus in this paper lies on a proof-of-concept for PF-based methods, we present the mean values across all files of a dataset, without analyzing correlations between accuracies and certain musical styles. We postpone such a more musical interpretation and its implications to a future publication. As the result of a PF is a random variable itself (due to the stochastic nature of a PF), we have carried out each experiment in the next section ten times and give the mean and the standard deviation across these ten trials for all PF approaches.

F-measure (FM)

The F-measure (FM) is computed from correctly detected beats within a window of ± 70 ms by

$$\text{F-measure} = \frac{2pr}{p+r} \tag{5.16}$$

where p (*precision*) denotes the ratio between correctly detected beats and all detected beats, and r ($recall$) denotes the ratio between correctly detected beats and the total number of annotated beats. The range of this measure is from 0% to 100%.

Allowed Metrical Level with no continuity required (AMLt)

In this method an estimated beat is counted as correct, if it lies within a small tolerance window around an annotated pulse, and the previous estimated beat lies within the tolerance window around the previous annotated pulse. The value of this measure is then the ratio between the number of correctly estimated beats divided by the number of annotated beats (as percentage between 0% and 100%). Beat sequences are also considered as correct if the beats occur on the off-beat, or are tapped at double or half the annotated tempo.

Information gain (InfG)

This measure is computed by calculating the timing errors between an annotation and all beat estimations within a one-beat length window around the annotation. Then, a beat error histogram is created from the resulting timing error sequence. A numerical score is derived by measuring the K-L divergence between the observed error histogram and the uniform distribution. This method gives a measure of how much information the beats provide about the annotations. The range of values for the Information Gain is 0 bits to approximately 5.3 ($= \log_2(40)$) bits, for 40 histogram bins.

Downbeat F-measure (DBFM)

For measuring the downbeat tracking performance, we use the same F-measure as for beat tracking (using a ± 70 ms tolerance window).

Runtime

The specified *runtimes* were measured using a MATLAB implementation of the systems on a PC with an Intel Core i5-2400 CPU with 3.1GHz. They include the computation of beats, downbeats and meter from the test dataset and exclude feature extraction and training which is the same for all models (e.g., for the Ballroom dataset feature extraction takes approximately 10 minutes and training the GMMs takes 30 seconds).

5.5.3 Determining system parameters

Both HMM and PF systems have a set of parameters that need to be determined. In the following we explain them in detail:

Observation model

For each test set, we learned the parameters of the observation model (see Section 5.3.4) from a non-overlapping training set, as shown in Table 5.1. This implies that for a given test set all PF and HMM methods use the same observation model.

Number of discrete states of the HMM

As explained in Section 5.4.1, the performance of the HMM depends on the density of the grid formed by the discretized variables. To visualize this dependency we illustrate the beat tracking accuracy and runtime of the HMM system for various state space sizes on the MBB training set in Fig. 5.6. In experiment 1, we will use two rhythmic patterns ($R = 2$, one for each of the two time signatures in the data), and in experiment 2, we will use eight rhythmic patterns ($R = 8$, one for each ballroom dance style). Considering both beat tracking accuracy and runtime, we chose to report results for three configurations of the HMM (depicted as HMM1-HMM3 in Table 5.2). Note that the largest model (HMM3) with $R = 2$ has a number of discrete states equal to $M \times N \times R = 209\,152$ and is therefore situated in the middle range of values depicted in Fig. 5.6.

Number of particles of the PFs

In all PF variants we used $N_S = 2\,000$ particles in experiment 1, and $N_S = 8\,000$ in experiment 2. As can be seen in Fig. 5.7, increasing the number of particles above this value did not further improve the performance. Also note the differences in runtime between AMPF and HMM by comparing Fig. 5.7 with Fig. 5.6, with a clear advantage for the PF which will be further documented in the next section on the larger evaluation datasets.

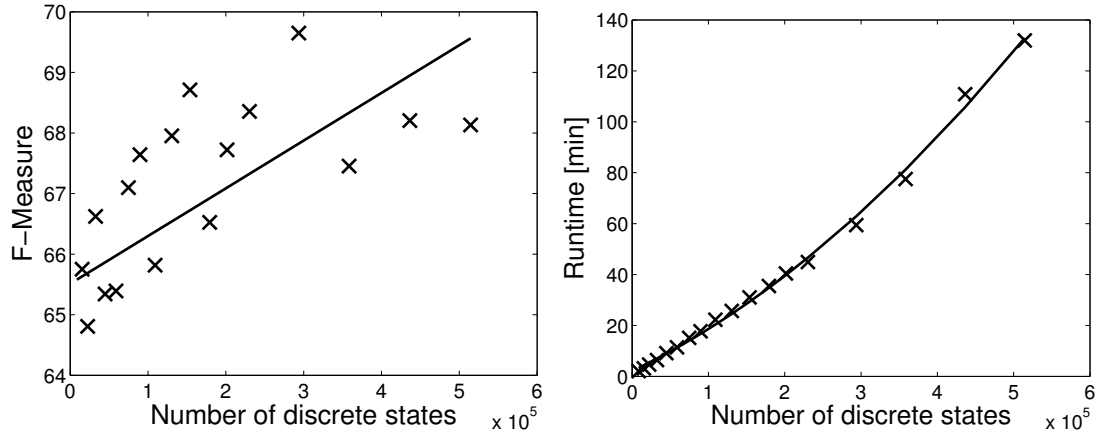


Figure 5.6: F-measure and runtime vs. number of discrete states for the HMM on the MBB set (total 41.9 minutes of audio).

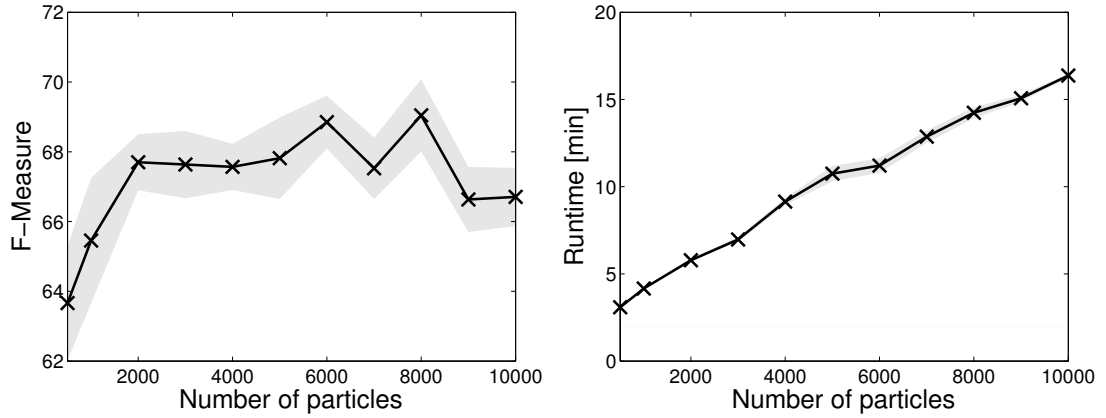


Figure 5.7: F-measure and runtime vs. number of particles for the AMPF on the MBB dataset (total 41.9 minutes). The obtained standard deviation across ten runs is depicted by the shaded gray area.

Apart from the design choices above, we obtained parameter values by performing a simple grid search over the parameter space using the MBB dataset and selecting the overall best performing parameters according to the three beat and one downbeat tracking measures. The determined parameters are:

- Tempo transition probability p_n of the HMM
- Resampling threshhold ρ of the SISR and APF approaches
- Parameters of the MPF/AMPF λ_ϕ, $\lambda_{\dot{\phi}}$, λ_r, τ_m, τ_s, d, β (see Section 5.4.2)

For the PFs, we repeated each experiment ten times and averaged the results. The selected parameters are shown in Table 5.2.

	ρ	d	$\sigma_{\dot{\phi}}$	λ_{ϕ}	$\lambda_{\dot{\phi}}$	λ_r	τ_m	τ_s	β
SISR	0.02	-	1.2	-	-	-	-	-	-
APF	0.1	-	1.2	-	-	-	-	-	1/5
MPF	-	30	1.2	1	1.4	1000	1	1.2	-
AMPF	-	30	1.2	1	1.4	1000	1	1.2	1/4
	M	N	p_n						
HMM1	640	12	0.02						
HMM2	1216	23	0.02						
HMM3	2432	43	0.02						

Table 5.2: Selected parameters for the experiments.

5.6 Experiments

5.6.1 Experiment 1: PF against HMM

Experiment 1 compares the four PF inference schemes (SISR, APF, MPF, AMPF) with the HMM inference, in terms of their runtime complexity, beat (all datasets) and downbeat (only *Ballroom* dataset) tracking accuracy. Tables 5.3 to 5.5 summarize the mean accuracies (and standard deviations for PF schemes) using all evaluation metrics on the three evaluation datasets.

	FM		AMLt		InfG		Runtime
	μ	σ	μ	σ	μ	σ	[minutes]
SISR	36.8	0.48	31.3	0.40	0.87	0.013	12.5
APF	38.7	0.46	32.8	0.64	0.91	0.014	19.0
MPF	38.8	0.48	32.7	0.90	0.89	0.022	17.8
AMPF	40.8	0.18	35.8	0.45	0.95	0.013	18.1
HMM1	39.6	-	31.7	-	0.87	-	11.1
HMM2	40.5	-	35.1	-	0.94	-	42.1
HMM3	**42.7**	-	**38.7**	-	**1.08**	-	164.0

Table 5.3: Beat tracking results and (average) runtimes on the SMC dataset (total 144.7 minutes). For the PF systems (SISR, APF, MPF, AMPF) we show the mean (μ) and the standard deviation (σ) over ten runs.

As a first conclusion, we can see that the four PF schemes improve in accuracy according to their ability to flexibly follow multi-modal distributions, with the proposed AMPF scheme showing superior performance compared to the other three schemes on all datasets and using all metrics. All standard deviations of the PF schemes are moder-

	FM		AMLt		InfG		Runtime
	μ	σ	μ	σ	μ	σ	[minutes]
SISR	62.4	0.43	72.3	0.32	1.93	0.0075	74.9
APF	64.6	0.33	74.9	0.35	1.98	0.0068	100.6
MPF	64.9	0.20	75.1	0.27	1.98	0.0059	115.7
AMPF	66.1	0.27	76.6	0.33	2.00	0.0063	120.1
HMM1	62.9	-	69.8	-	1.73	-	69.0
HMM2	65.4	-	76.2	-	1.94	-	264.0
HMM3	**67.4**	-	**79.9**	-	**2.09**	-	1016.4

Table 5.4: Beat tracking results and (average) runtimes on the 1360-song dataset (total 861.6 minutes). For the PF systems (SISR, APF, MPF, AMPF) we show the mean (μ) and the standard deviation (σ) over ten runs.

ate, which indicates that their performance can be expected to be reliable throughout individual evaluations. Comparing the best particle filtering inference scheme, AMPF, with the three HMM parametrizations, it is apparent that the largest HMM3 slightly outperforms the AMPF on all three datasets. However, this comes at a high price in terms of runtimes, as can be seen from the rightmost columns in Tables 5.3 to 5.5. This result was expected, since it confirms that the HMM on a sufficiently dense grid is able to perform accurate inference that cannot be outperformed using approximate methods (such as the PF) using the same underlying model. Comparing the performance of HMM3 and AMPF in the *SMC* and *1360-song* datasets with the performances of other algorithms on the same dataset (Holzapfel et al., 2012), it becomes apparent that the HMM3 generally outperforms all other approaches, while the AMPF simply performs as good as the most performing systems. This finding implies that the underlying bar pointer model is a relatively accurate model for meter inference in music in comparison to the state of the art (for the *SMC* and *1360-song* datasets), and the AMPF represents a fast and accurate approximation to the best performance of HMM3. In the following experiment we will evaluate the potential of increasing the number of rhythmic patterns.

5.6.2 Experiment 2: Increasing pattern diversity

In the *second experiment* we enlarge the state space by introducing eight style specific rhythmic pattern states into the model, one for each dance style in the *Ballroom* dataset (due to the low number of samples we merged all three Rumba dance styles into one Rumba category). The resulting eight rhythmic patterns are learned on the *Ballroom_train* subset using the dance style labels that come with the data. We compare

	FM		AMLt		InfG		DBFM		Runtime
	μ	σ	μ	σ	μ	σ	μ	σ	[minutes]
SISR	74.5	0.99	84.8	0.94	2.39	0.034	37.8	2.11	8.7
APF	76.9	0.88	87.5	0.18	2.49	0.017	45.0	1.32	10.9
MPF	82.7	0.70	89.9	0.50	2.52	0.020	53.4	1.19	12.7
AMPF	83.6	0.46	90.5	0.23	2.52	0.012	55.8	1.56	13.7
HMM1	77.5	-	83.0	-	2.11	-	54.9	-	8.1
HMM2	83.2	-	90.2	-	2.49	-	60.6	-	28.9
HMM3	**85.1**	-	**92.1**	-	**2.68**	-	**63.3**	-	111.8

Table 5.5: Beat and downbeat tracking results and average runtimes on the Ballroom_test dataset (total 106.4 minutes). For the PF systems (SISR, APF, MPF, AMPF) we show the mean (μ) and the standard deviation (σ) over ten runs.

the beat and downbeat tracking accuracy of the PF and HMM inference methods applied to this enlarged state space in Table 5.6. For the MPF, AMPF, and HMM methods, and all evaluation measures except AMLt a clear performance improvement over the accuracies depicted in Table 5.5 can be seen. The reason for stagnation in the AMLt is easy to explain: The inclusion of more accurate rhythmic patterns leads to less tempo halving or doubling errors, and to less off-beat estimations in Table 5.6. While such an improvement can be important in certain applications such as transcription or chord estimation, it does not affect AMLt. Furthermore, the SISR and APF seem to have difficulties to handle the enlarged state-space, as indicated by a drastic decrease of beat tracking performance. In contrast, the MPF, AMPF, and HMMs seem to benefit from the more precise model. We can therefore conclude that a more precise modeling of the rhythmic style in a collection (by using a higher number of rhythmic patterns) has the potential to further increase the performance of our model. However, this might be no longer feasible using the HMM with the highest resolution. The HMM3 takes about three times real time to process the data as shown in the rightmost column of Table 5.6. Therefore, the inference using a model with several rhythmical pattern states marks the point where approximate inference with PFs becomes necessary, at least with the computational power at the time of writing this paper.

5.7 Discussion

Our results on the largest available annotated datasets support that the bar pointer model described in Section 5.3 is a relatively accurate model for inferring metrical structure from (metered) musical audio signals. Using an exact inference scheme (HMM) in a densely sampled discretized space, we achieve beat tracking accuracies that outper-

	FM		AMLt		InfG		DBFM		Runtime
	μ	σ	μ	σ	μ	σ	μ	σ	[minutes]
SISR	66.0	0.94	72.4	1.11	2.23	0.044	39.6	2.14	26.9
APF	70.2	1.26	76.0	1.05	2.37	0.025	46.4	2.07	43.2
MPF	88.3	1.04	90.1	0.52	2.79	0.028	63.9	1.41	43.8
AMPF	89.3	0.54	90.9	0.22	2.78	0.014	67.6	1.22	44.9
HMM1	85.5	-	87.5	-	2.39	-	68.1	-	24.0
HMM2	89.4	-	90.3	-	2.70	-	72.1	-	87.2
HMM3	**90.5**	-	**91.9**	-	**2.90**	-	**73.5**	-	337.3

Table 5.6: Beat and downbeat tracking results and average runtimes on the Ballroom_test dataset (total 106.4 minutes) using eight rhythmic pattern states. For the PF systems (SISR, APF, MPF, AMPF) we show the mean (μ) and the standard deviation (σ) over ten runs.

form those documented for the best state-of-the-art approaches (comparing Tables 5.3 and 5.4 with Tables I and II in Holzapfel et al. (2012)). The approximate AMPF scheme still achieves accuracies as high as the best state-of-the-art approaches, slightly inferior to the best evaluated HMM. However, good performance with the HMM inference comes at a high price. We need to use HMMs with a large state space (HMM3 in our experiments), which becomes even larger for experiments with several rhythmic pattern states, as in our experiment on the *Ballroom* dataset (Table 5.6).

Furthermore, our results indicate that modeling several rhythmic patterns improves the performance for music collections with diverse rhythmical content, at least in music with limited expressiveness as investigated in this paper. However, with the resulting enlargement of the discretized state space the usage of a HMM becomes computationally prohibitive and our proposed AMPF is a fast and accurate alternative.

The experiments show that the AMPF scheme (and to some extend also the MPF) handles the degeneracy problem much better than other PF methods because it maintains the diverse multi-modal probability distribution that is crucial for the tracking of multiple tempo modes that occur in music.

In Fig. 5.8 we demonstrate for an exemplary audio file that the proposed AMPF and MPF, in contrast to the APF and SISR, are able to track the multiple modes of the posterior (light gray regions in Fig. 5.8) throughout a recording. The figure shows that both SISR and APF concentrate their particles on a few modes of the posterior after only five seconds of the recording (see bottom row of Fig. 5.8). In contrast, the AMPF is characterized by the most diverse particle distribution throughout a song. This diversity depicted for the example is typical for the approach, and is the reason for the improved performance of the AMPF compared to the other PF schemes.

It is worth to point out that the results reported in this paper might be further improved if the rhythmic patterns were learned from music that is rhythmically more similar to the test set. For instance, this could be achieved by downbeat annotating a smaller subset of a certain style (e.g., the Greek music samples present in the *1360-song* collection), and then attempting to track the meter in the other Greek music samples. Nevertheless, in order to adapt our model to a certain style, a representative set of songs has to be beat and downbeat annotated. However, apart from such annotation work, no changes to the structure of the model or its inference need to be performed.

5.8 Conclusion

In this paper we presented for the first time a particle filtering scheme for beat tracking in music that takes into account multiple modes of the posterior probability, along with a systematic evaluation on larger datasets. The comparison with the HMM inference demonstrates its superiority in terms of computational load, while the accuracy of the system is as high as the state-of-the-art in beat tracking. While the presented results prove the applicability of the approach, we need to do further steps in modeling style specific rhythmic patterns in order to better understand the true potential of the method.

Perhaps most importantly, we can claim that the separation of observation model and the hidden variables causes a decoupling of the internal inference from the actual musical sound. This means that the model can potentially be adapted without parameter tweaking or engineering knowledge to new musical styles. By means of this design, the presented method avoids a systematic bias that can result from a hard-coding of music properties into system parameters. It represents a method that can be used for style specific estimation in a straight-forward way by annotating a representative music corpus. Therefore, the system is consistent with the demands of the recent MIR roadmap (Serra et al., 2013) for systems which are able to incorporate expert knowledge.

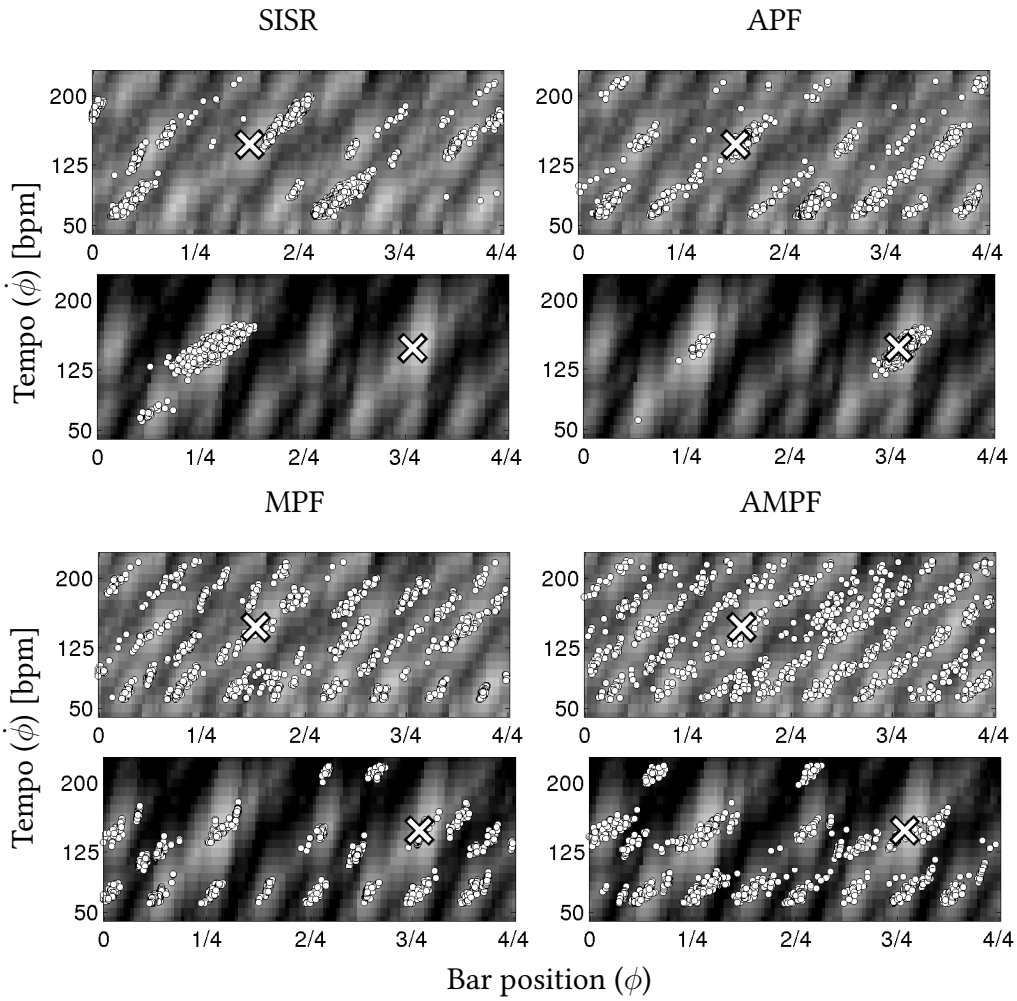


Figure 5.8: Particle locations (white dots) within the state-space (tempo and bar position sub-space) after one second (top row) and after five seconds (bottom row) of the song *Lemon Tree* by *Fool's Garden* plotted on top of the (log) posterior probability computed by the HMM. Light gray tones indicate a high posterior probability while dark gray tones indicate a low probability. The cross marks the groundtruth (bar position and tempo for the corresponding time points).

Chapter 6

An Efficient State Space Model for Joint Tempo and Meter Tracking

Published In Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR) (Krebs et al., 2015a).

Authors Florian Krebs, Sebastian Böck, and Gerhard Widmer.

Personal contributions The idea to make the state space more compact arose in discussions with Sebastian. I implemented it in MATLAB for the paper, Sebastian later integrated it into the *madmom* package (Böck et al., 2016a).

Abstract Dynamic Bayesian networks (e.g., Hidden Markov Models) are popular frameworks for meter tracking in music because they are able to incorporate prior knowledge about the dynamics of rhythmic parameters (tempo, meter, rhythmic patterns, etc.). One popular example is the *bar pointer model*, which enables joint inference of these rhythmic parameters from a piece of music. While this allows the mutual dependencies between these parameters to be exploited, it also increases the computational complexity of the models. In this paper, we propose a new state-space discretisation and tempo transition model for this class of models that can act as a drop-in replacement and not only increases the beat and downbeat tracking accuracy, but also reduces time and memory complexity drastically. We incorporate the new model into two state-of-the-art beat and meter tracking systems, and demonstrate its superiority to the original models on six datasets.

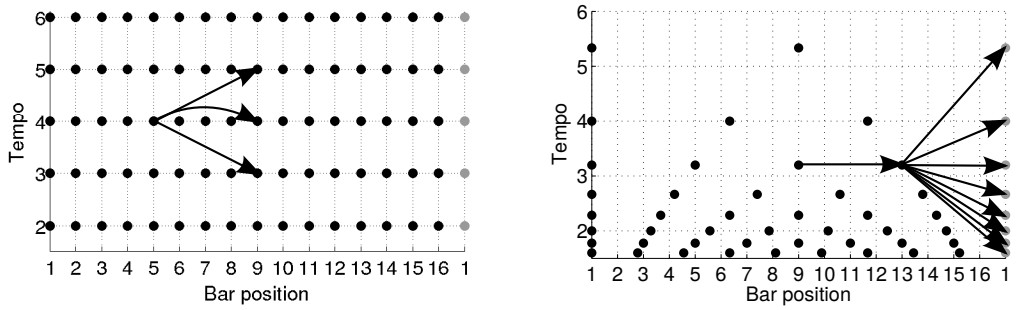
6.1 Introduction

Building machines that mimic the human understanding of music is vital for a variety of tasks, such as organising and managing today's huge music collections. In this context, automatic inference of metrical structure from a musical audio signal plays an important role. Generally, the metrical structure of music builds upon a hierarchy of approximately regular pulses with different frequencies. In the centre of this hierarchy is the *beat*, a pulse to which humans choose to tap their feet. These beats are again grouped into bars, with the *downbeat* denoting the first beat of each bar.

Several approaches have been proposed for tackling the problem of automatic inference of meter (or subcomponents such as beats and downbeats) from an audio signal, with approaches based on machine learning currently being the most successful(Zapata et al., 2014; Korzeniowski et al., 2014; Böck et al., 2014; Holzapfel et al., 2014; Durand et al., 2015). All of these approaches incorporate probabilistic models, but with different model structures: the systems introduced by Zapata et al. (2014); Korzeniowski et al. (2014); Durand et al. (2015) decouple tempo detection from the detection of the beat/downbeat phase, which has the advantage of reducing the search space of the algorithms but can be problematic if the tempo detection is erroneous. Others (Böck et al., 2014; Holzapfel et al., 2014) model tempo and beat/downbeat jointly, taking into account their mutual dependency, which leads to increased model complexity.

One popular model that jointly models tempo and bar position is the *bar pointer model*, first proposed by Whiteley et al. (2006). In addition to tempo and bar position, the model also integrates various rhythmic pattern states. It has been extended by various authors: Krebs et al. (2013); Holzapfel et al. (2014) demonstrated the benefit of using rhythmic pattern states to analyse rhythmically diverse music, Srinivasamurthy et al. (2015) proposed a simplification for models with multiple rhythmic pattern states, Şimşekli et al. (2012) additionally modelled the label of an acoustic event in order to enable a drum robot to distinguish different instruments, and Dzhambazov (2014) applied it to a drum transcription task. These algorithms share the problem of a high space and time complexity because of the huge state-space in which they perform inference. In order to make inference tractable, the state-space is usually divided into discrete cells, with either fixed (Whiteley et al., 2006; Krebs et al., 2013; Holzapfel et al., 2014; Böck et al., 2014; Şimşekli et al., 2012; Dzhambazov, 2014) or dynamic (Whiteley et al., 2007; Krebs et al., 2015b; Srinivasamurthy et al., 2015) locations in the state-space. While the former approach can be formulated as a hidden Markov model (HMM), which performs best but is prohibitively complex, the latter uses particle filtering (PF), which is fast but performs slightly worse in sub-tasks such as downbeat tracking (Krebs et al., 2015b).

In this paper, we propose a modified bar pointer model which not only increases beat and downbeat tracking accuracy, but also reduces drastically time and memory complexity. In particular, we propose (a) a new (fixed grid) discretisation of the joint



(a) Original discretisation (Whiteley et al., 2006)

(b) Proposed discretisation

Figure 6.1: Toy example with $M = 16$ and $N = 6$: Each dot corresponds to a (hidden) state in the tempo-bar-position state-space. The arrows indicate examples of possible state transitions.

tempo and beat/bar state-space and (b) a new tempo transition model. We incorporated the new model into two state-of-the-art beat and meter tracking systems, and demonstrate its superiority on six datasets.

6.2 Method

In this section, we describe how we tackle the problem of metrical structure analysis using a probabilistic state-space model. In these models, a sequence of *hidden variables*, which in our case represent the meter of an audio piece, is inferred from a sequence of *observed variables*, which are extracted from the audio signal. For ease of presentation, we now consider a state-space of two hidden variables, the position within a bar and the tempo. Including additional hidden variables, e.g., a rhythmical pattern state (Whiteley et al., 2006, 2007; Krebs et al., 2013; Holzapfel et al., 2014; Srinivasamurthy et al., 2015) or an acoustic event label (Şimşekli et al., 2012; Dzhambazov, 2014) is straightforward. In the following, we describe the original bar pointer model (Whiteley et al., 2006), its shortcomings, and the proposed improvements.

6.2.1 The original bar pointer model

The bar pointer model (Whiteley et al., 2006) describes the dynamics of a hypothetical pointer which moves through the space of the hidden variables throughout a piece of music. At each time frame k, we refer to the (hidden) state of the bar pointer as $\mathbf{x}_k = [\Phi_k, \dot{\Phi}_k]$, with $\Phi_k \in \{1, 2, ..., M\}$ denoting the position within a bar, and $\dot{\Phi}_k \in \{\dot{\Phi}_{min}, \dot{\Phi}_{min} + 1, ..., \dot{\Phi}_{max}\}$ the tempo in bar positions per time frame. M is the total

number of discrete positions per bar, $N = \dot{\Phi}_{max} - \dot{\Phi}_{min} + 1$ is the total number of distinct tempi, $\dot{\Phi}_{min}$ and $\dot{\Phi}_{max}$ are respectively the lowest and the highest tempo. See Fig. 6.1a for an illustration of such a state space. Finally, we denote the observation features as $\mathbf{y}_k$.

Overall, we want to compute the most likely hidden state sequence $\mathbf{x}^*_{1:K} = \{\mathbf{x}^*_1, \mathbf{x}^*_2, ..., \mathbf{x}^*_K\}$ given a sequence of observations $\{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_K\}$ for each audio piece as

$$\mathbf{x}^*_{1:K} = \arg\max_{x_{1:K}} P(\mathbf{x}_{1:K} \mid \mathbf{y}_{1:K}). \tag{6.1}$$

with

$$P(\mathbf{y}_{1:K}|\mathbf{x}_{1:K}) \propto P(\mathbf{x}_1) \prod_{k=2}^{K} P(\mathbf{x}_k|\mathbf{x}_{k-1}) P(\mathbf{y}_k|\mathbf{x}_k). \tag{6.2}$$

Here, $P(\mathbf{x}_1)$ is the *initial state distribution*, $P(\mathbf{x}_k|\mathbf{x}_{k-1})$ is the *transition model*, and $P(\mathbf{y}_k|\mathbf{x}_k)$ is the *observation model*, which we further describe in the bottom of this section. Eq. 6.1 can be solved using the well-known Viterbi algorithm (Rabiner, 1989). Finally, the set of downbeat frames $\mathcal{D}$ can be extracted from the sequence of bar positions as

$$\mathcal{D} = \{k : \Phi^*_k = 1\}, \tag{6.3}$$

and the set of beat frames can be obtained analogously by selecting the time frames which correspond to a bar position that matches a beat position.

Initial distribution

Here, any prior knowledge (e.g., about tempo distributions) can be incorporated into the model. Like most systems, we use a uniform distribution in this work.

Transition model

The transition model $P(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ can be further decomposed into a distribution for each of the two hidden variables Φ_k, and $\dot{\Phi}_k$ by:

$$\begin{aligned} P(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \quad & P(\Phi_k \mid \Phi_{k-1}, \dot{\Phi}_{k-1}) \cdot \\ & \cdot P(\dot{\Phi}_k \mid \dot{\Phi}_{k-1}). \end{aligned} \tag{6.4}$$

The first factor is

$$P(\Phi_k \mid \Phi_{k-1}, \dot{\Phi}_{k-1}) = \mathbb{1}_x, \tag{6.5}$$

where $\mathbb{1}_x$ is an indicator function that equals one if $\Phi_k = (\Phi_{k-1} + \dot{\Phi}_{k-1} - 1) \bmod M + 1$, and zero otherwise. The modulo operator makes the bar position cyclic (the last, light grey column in Fig. 6.1a is identical to the first column).

The second factor $P(\dot{\Phi}_k \mid \dot{\Phi}_{k-1})$ is implemented by
If $\dot{\Phi}_{min} \leq \dot{\Phi}_k \leq \dot{\Phi}_{max}$,

$$P(\dot{\Phi}_k \mid \dot{\Phi}_{k-1}) = \begin{cases} 1 - p_{\dot{\Phi}}, & \dot{\Phi}_k = \dot{\Phi}_{k-1}; \\ \frac{p_{\dot{\Phi}}}{2}, & \dot{\Phi}_k = \dot{\Phi}_{k-1} + 1; \\ \frac{p_{\dot{\Phi}}}{2}, & \dot{\Phi}_k = \dot{\Phi}_{k-1} - 1, \end{cases} \tag{6.6}$$

otherwise $P(\dot{\Phi}_k|\dot{\Phi}_{k-1}) = 0$.
$p_{\dot{\Phi}}$ is the probability of a tempo change. From Eq. 6.6 it can be seen that the pointer can perform three tempo transitions from each state (indicated by arrows in Fig. 6.1a).

Observation model

In this paper, we use two different observation models: The first one uses *recurrent neural networks* to derive a probability of a frame being a beat or not (Böck et al., 2014). The second one models the observation probabilities with Gaussian mixture models from a two-dimensional onset feature (Krebs et al., 2013; Holzapfel et al., 2014). As the focus of this paper lies on the state discretisation and the tempo transition model, the reader is referred to (Krebs et al., 2013; Holzapfel et al., 2014; Böck et al., 2014) for further details.

6.2.2 Shortcomings of the original model

Previous implementations of the bar pointer model (Şimşekli et al., 2012; Krebs et al., 2013; Holzapfel et al., 2014; Böck et al., 2014; Collins et al., 2014; Dzhambazov, 2014) followed Whiteley et al. (2006) in dividing the tempo-position state space into equidistant points, with each point aligned to an integer-valued bar position and tempo (see Fig. 6.1a). This discretisation has a number of drawbacks, which are further explained in the following.

Time resolution

As shown in Fig. 6.1a, the number of position grid points per bar is constant across the tempi. This means that the grid of a bar played at a low tempo has a lower time resolution than of a bar played at high tempo, because both are divided into the same number of cells. In contrast, there are more observations available for a bar at a low tempo than for a bar at a high tempo, since the observations are extracted at a constant frame rate. This causes a mismatch between the time resolution of the feature extraction and the time resolution of the discretised bar position.

Tempo resolution

As shown in Fig. 6.1a, the distance between two adjacent tempo grid points is constant across the grid. This is inconsistent with tempo sensitivity experiments on humans, which have shown that the human ability to notice tempo changes is proportional to the tempo, with the JND (just noticeable difference) being around 2-5% of the inter beat interval (Drake and Botte, 1993). Therefore, in order to get a sufficiently high tempo resolution at lower tempi, a huge number of tempo states has to be chosen.

Tempo stability

As the tempo model (see Eq. 6.6) forms a first-order Markov chain, the current tempo state is independent of all tempo states given the past tempo state. This means that the tempo model is not able to reflect any long term dependencies between tempo states, which may result in unstable tempo trajectories.

6.2.3 Proposed model

This section introduces a solution to the problems described above. To simplify notation we assume a bar has four beats. Extending to other time signatures (Whiteley et al., 2006) or modelling beats instead of bars (Böck et al., 2014) is straightforward.

Time resolution

We propose making the number of discrete bar positions M dependent on the tempo by using exactly one bar position state per audio frame (and thus per observation feature value). The number of observations per bar (four beats) at a tempo T in beats per minute (BPM) is

$$M(T) = \text{round}(\frac{4 \times 60}{T * \Delta}) \tag{6.7}$$

with Δ being the audio frame length. Using Eq. 6.7, we compute the number of bar positions of the tempo limits $M(T_{min})$ and $M(T_{max})$.

Tempo resolution

We can now either model all N_{max} tempi that correspond to integer valued bar positions in the interval $[M(T_{max}),\, M(T_{min})]$, with

$$N_{max} = M(T_{min}) - M(T_{max}) + 1, \tag{6.8}$$

or select only a subset of N tempo states. In Section 6.3, we evaluate the performance of the transition model for various numbers of tempo states. For $N < N_{max}$, we choose the tempo states by distributing N states logarithmically across the range of

beat intervals, trying to mimic the JNDs of the human auditory system (Drake and Botte, 1993).

Tempo stability

To increase the stability of the tempo trajectories we only allow transitions at beat positions within a bar. This is illustrated in Fig. 6.1b with the arrows showing examples of possible state transitions. In contrast to the original model which allows three tempo transitions at every time step, we allow transitions to each tempo, but only at beat times. The new tempo transition model then becomes:
If $\Phi_k \in \mathcal{B}$,

$$P(\dot{\Phi}_k|\dot{\Phi}_{k-1}) = f(\dot{\Phi}_k, \dot{\Phi}_{k-1})$$

else

$$P(\dot{\Phi}_k|\dot{\Phi}_{k-1}) = \begin{cases} 1, & \dot{\Phi}_k = \dot{\Phi}_{k-1}; \\ 0, & \text{otherwise} \end{cases} \tag{6.9}$$

$\mathcal{B}$ is the set of bar positions that corresponds to beats, and $f(\cdot)$ is a function that models the tempo change probabilities. We experimented with various functions (Gaussian, Log-Gaussian, Gaussian mixtures), but found this exponential distribution to be performing best:

$$f(\dot{\Phi}_k, \dot{\Phi}_{k-1}) = exp(-\lambda \times |\frac{\dot{\Phi}_k}{\dot{\Phi}_{k-1}} - 1|) \tag{6.10}$$

where the rate parameter $\lambda \in \mathbb{Z}_{\geq 0}$ determines the steepness of the distribution. A value of $\lambda = 0$ means that transitions to all tempi are equally probable. In practice, for music with roughly constant tempo, we set $\lambda \in [1, 300]$. Fig. 6.2 shows the tempo transition probabilities for various values of λ.

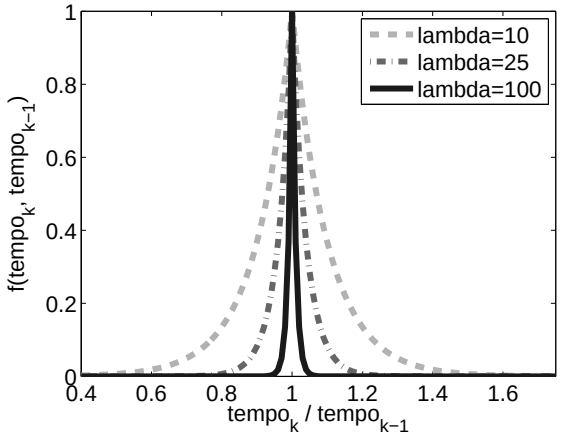


Figure 6.2: Tempo change probability density (Eq. 6.10) for various values of λ.

6.2.4 Complexity of the inference algorithm

In this section, we investigate time and memory complexity of the bar pointer model, considering only the complexity of the (Viterbi) inference and ignoring the contribution of computing the observation features and observation probabilities.

Both time and space complexity depend on the number of states of the model. The number of states, in turn, depends on the number of bar positions, the tempo ranges, the audio frame length, and the tempo resolution that we chose to model. Let us assume that we have a model with S hidden states, T possible state transitions per frame, and an audio excerpt with K frames. The memory requirement of the algorithm is then simply $S \times K$, as we have to store the best predecessor state for each of the S states for each time frame during Viterbi decoding. The time complexity, on the other hand, is $T \times K$, as we have to compute T transitions at each time step. In Table 6.1 we show the values of S and T of the models used in this paper.

6.3 Experimental setup

In this section, we evaluate the proposed model with real-world music data in two experiments[1]. In the first experiment, we investigated the effect of the number of tempo states N and the rate parameter λ of the tempo transition function on the meter tracking performance on a training set. We evaluated only the beat tracking performance, as this is the most fundamental task that we wanted to solve. In the second experiment, we integrated the proposed model with the parameters determined in Experiment 1 into two state-of-the-art systems and compared the meter tracking performance in terms of accuracy and complexity with the original models. Below, we describe the datasets, the evaluation metrics, and the meter tracking models.

6.3.1 Datasets

In this work, we used seven test datasets, one for Experiment 1 and the remaining six for Experiment 2. For more details about each datasets, see the corresponding references:

Experimental dataset: This dataset is a subset of the *1360-songs* dataset (Gouyon, 2005) excluding the Hainsworth dataset, because it was used in Experiment 2. In total, it includes 1139 excerpts (total length 662 minutes).

Ballroom dataset (Gouyon et al., 2006): A dataset of 685 30-second excerpts of ballroom dance music (total length 364 minutes). It was annotated with beat and downbeat times by Krebs et al. (2013).

[1] Additional information as well as the code to reproduce the results of this paper are available at http://www.cp.jku.at/people/krebs/ismir2015/

Hainsworth dataset (Hainsworth and Macleod, 2004): A dataset with 222 pieces (total length 199 minutes), covering a wide spectrum of genres.

SMC dataset (Holzapfel et al., 2012): A dataset with 217 pieces which are considered difficult for meter inference (total length 145 minutes). This set is also part of the MIREX evaluation.

Greek dataset (Holzapfel et al., 2014): 42 full songs of Cretan leaping dances in $2/4$ meter (total length 140 minutes).

Turkish dataset (Holzapfel et al., 2014): 82 one-minute excerpts of Turkish Makam music (total length 82 minutes).

Indian dataset (Srinivasamurthy and Serra, 2014): The same subset of 118 two-minute long pieces (total length 235 minutes) as used by Holzapfel et al. (2014).

6.3.2 Evaluation metrics

To assess the ability of an algorithm to infer metrical structure, we used five evaluation metrics - four for beat tracking and one for downbeat tracking.

F-Measure (FM): computed from the number of true positives (correctly detected beats within a window of ± 70ms around an annotation), the false positives, and the false negatives.

CMLt: quantifies the percentage of correctly tracked beats at the correct metrical level. In order to count a beat as correct, both previous and next beats have to match an annotation within a tolerance window of $\pm 17.5\%$ of the annotated beat interval.

AMLt: the same as CMLt, but the detected beats are also considered to be correct if they occur on the off-beat or at double or half of the ground-truth tempo.

Cemgil: places a Gaussian function with standard deviation of 40 ms around the annotations and computes the average likelihood of the corresponding beat closest to each annotation. In contrast to the other measures with hard decision boundaries (due to rectangular tolerance windows), this measure is also sensitive to small timing differences between annotated and detected beats.

Downbeat F-Measure (DBFM): is the same F-measure as used for beats, but considers only downbeats.

We implemented the evaluation metrics according to Davies et al. (2009) with standard settings. To make them comparable with other work, we excluded the first five seconds in Experiment 2 when comparing with the model by Holzapfel et al. (2014) but did not exclude them when comparing with the results from (Böck et al., 2014).

6.3.3 Meter tracking models

To compare the proposed to the original model, we tested its performance with two state-of-the-art meter tracking systems:

RNN-BeatTracker (Böck et al., 2014): This model uses a *recurrent neural network* to compute the probability of a frame being a beat. This probability is used as an observation probability for an HMM which jointly models tempo and the position within a beat period. We used the same *MultiModelBeatTracker* model as described by Böck et al. (2014), with a frame rate of 100 fps. Only beats are detected with this model.

GMM-BarTracker (Krebs et al., 2013; Holzapfel et al., 2014): Gaussian Mixture Models (GMMs) are used to compute the observation probabilities for an HMM that jointly models tempo, position within a bar and a set of rhythmic bar-patterns. For Experiment 1, the GMMs were trained on the *Ballroom*, the *Beatles* (Davies et al., 2009), the *Hainsworth* and the *RWC_Popular* (Goto, 2006) datasets, using three rhythmic patterns that correspond to the time signatures $2/4, 3/4$ and $4/4$. Pieces with other time signatures were excluded. For Experiment 2, we used an updated[2] version of the model described by Holzapfel et al. (2014). The model uses a frame length of 20 ms and integrates eight rhythmic pattern states, one for each of the rhythmic classes. It outputs beats and downbeats.

Note that the difference between *original* and *proposed* lies only in the definition of the hidden states and the transition model; both use the same observation model, initial distribution, and tempo ranges.

6.4 Results and discussion

6.4.1 Experiment 1

In this experiment, we evaluated the influence of two parameters of the proposed transition model on the meter tracking performance. These parameters are the width of the tempo change distribution parametrised by the rate λ (Section. 6.2.3, Fig. 6.3) and the number of tempo states N (Section 6.2.3, Fig. 6.4). We chose to display the *Cemgil* accuracy in Figs. 6.3 and 6.4, because it is the only measure that makes a soft decision to count a beat as correct by using a Gaussian window and thus also takes into account small timing variations. Generally, the plots for the other measures were similar.

Fig. 6.3 shows the effect of the parameter λ on the *Cemgil* beat tracking accuracy for both the *RNN-BeatTracker* and the *GMM-BarTracker* on the *experimental* dataset, using the maximum number of tempo states N_{max}. The maximum *Cemgil* values were obtained with $\lambda = 125$, and $\lambda = 95$ respectively.

Using these settings for λ, we investigated the effect of the number of tempo states N on the beat tracking performance, which is shown in Fig. 6.4. As the two systems use a different audio frame rate, the maximum number of tempo states N_{max} is different too (see Section 6.2.3). Using a tempo range of $[55, 215]$ BPM as in (Böck et al., 2014), the *RNN-BeatTracker* has at most $N_{max} = 82$ tempo states, while for the *GMM-BarTracker*

[2] http://www.cp.jku.at/people/krebs/ismir2014/

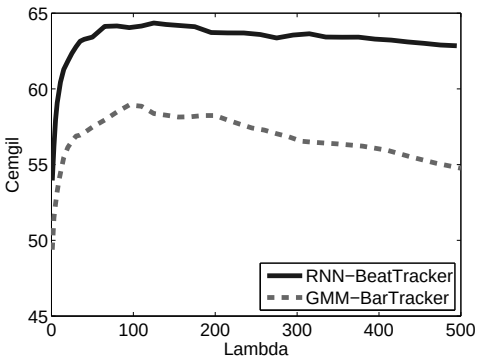


Figure 6.3: Effect of parameter λ on beat tracking *Cemgil* metric on the *experimental* dataset.

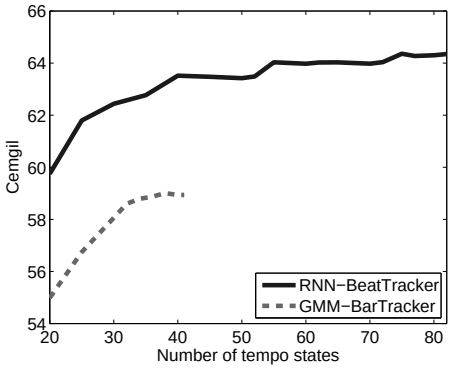


Figure 6.4: Effect of the number of tempo states on beat tracking *Cemgil* metric on the *experimental* dataset.

$N_{max} = 41$. As can be seen from Fig. 6.4, the *Cemgil* accuracy converges at ≈ 75 tempo states for the *RNN-BeatTracker* and at ≈ 40 for the *GMM-BarTracker*. This finding suggests that the *BarTracker* might also benefit from a higher audio frame rate and therefore a higher number of tempo states. In addition, the number of tempo states is a suitable parameter to select a trade-off between speed and accuracy.

6.4.2 Experiment 2

In this experiment, we integrated the proposed model into two state-of-the-art meter tracking systems (Section 6.3.3) and compared them to the original models. The beat and downbeat accuracy scores of the original (Böck et al., 2014; Holzapfel et al., 2014) and proposed models, together with the number of states and transitions, are shown in Table 6.1. The proposed model used the parameters λ and N obtained in Experiment 1.

As can be seen, the proposed transition model outperforms the original model with respect to all performance metrics on all datasets (except *AMLt* (-0.2%) on the Ballroom dataset), with the added advantage of drastically reduced complexity. The *CMLt* metric in particular seems to benefit from the proposed model, with up to 20% relative improvement on the *Greek* dataset. Apparently, the restriction to change tempo only at beat times results in higher stability and therefore better performance in measures that are sensitive to continuity, such as *CMLt* and *AMLt*.

A comparison of the state-space sizes of the original and proposed models shows that the latter uses far fewer states and transitions. This is particularly apparent for the *GMM-BarTracker*, which has *a priori* a larger state space because it models (a) bars instead of beats and (b) eight rhythmic patterns. With the original *GMM-BarTracker*,

	# tempo states	FM	CMLt	AMLt	DBFM	States	Transitions
RNN-BeatTracker							
Ballroom							
Böck et al. (2014)	20	91.0	83.0	92.4		11 520	33 280
Proposed	82	91.9	85.4	92.2		5 617	8 343
Proposed	55	91.7	84.8	92.1		3 369	4 496
Hainsworth							
Böck et al. (2014)	20	84.0	80.3	88.1		11 520	33 280
Proposed	82	85.1	80.5	88.5		5 617	8 343
Proposed	55	85.1	79.1	88.6		3 369	4 496
SMC							
Böck et al. (2014)	20	52.9	42.8	56.7		11 520	33 280
Proposed	82	54.0	46.0	61.3		5 617	8 343
Proposed	55	54.3	45.8	61.3		3 369	4 496
GMM-BarTracker							
Greek							
Holzapfel et al. (2014)	18	91.6	77.8	95.2	77.7	133 200	376 800
Proposed	35	95.6	93.5	+96.5	81.2	26 716	41 708
Indian							
Holzapfel et al. (2014)	18	79.9	61.3	84.5	47.6	133 200	376 800
Proposed	35	85.0	+70.3	94.2	+ 51.5	26 716	41 708
Turkish							
Holzapfel et al. (2014)	18	86.1	69.4	84.0	61.7	133 200	376 800
Proposed	35	87.7	73.2	87.7	63.2	26 716	41 708

Table 6.1: Performance of the original and proposed transition model and state space. The + symbol denotes significant ($p < 0.05$) improvement over the result in the row above, using a one-way analysis of variance (ANOVA) test.

processing a four-minute piece ($12\,000$ frames at 50 fps), required remembering 1.60×10^9 state ids in the Viterbi algorithm, which needs 6.39 GB stored as 32-bit integers. In contrast, using the proposed model, only 0.32×10^9 states must be stored - a demand that can be met using 16-bit integers in only 0.64 GB of memory. With a MATLAB implementation and an Intel Core i5-2400 CPU with 3.1 GHz, we can therefore reduce the computation time for the *Turkish* dataset from 45.8 minutes to 4.2 minutes, including the computation of the audio features (which takes only 18 seconds). Additionally, as already shown in Experiment 1, we can further reduce the number of tempo states from 82 (the maximum number of tempo states as computed in Section 6.2.3) to 55 with the *RNN-BeatTracker*, with only marginal performance decrease. Compared to the original model, this implies a reduction of the numbers of states and transitions by factors of three and seven, respectively. Since in the proposed model most position states are needed to model lower tempi, the lower tempo limits mainly determine the size of the state space.

6.5 Conclusions

In this paper, we have proposed a new discretisation and tempo transition model that can be used as a drop-in replacement for variants of the *bar pointer model*. We have shown that our model outperformed the original one in 32 of 33 test cases, while substantially reducing space and time complexity. We believe that this is an important step towards lightweight, real-time capable, high-performance meter inference systems.

As part of future work, we plan to investigate whether changing tempo only at beat positions also stabilises the particle filter versions of the *bar pointer* model (Krebs et al., 2015b; Srinivasamurthy et al., 2015), which would further facilitate reducing computational complexity.

Chapter 7

Downbeat Tracking Using Beat Synchronous Features and Recurrent Neural Networks

Published In Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR) (Krebs et al., 2016).

Authors Florian Krebs, Sebastian Böck, Matthias Dorfer and Gerhard Widmer.

Personal contributions Matthias and Sebastian gave me valuable advice on training the RNNs. I did the implementations and ran the experiments.

Abstract In this paper, we propose a system that extracts the downbeat times from a beat-synchronous audio feature stream of a music piece. Two recurrent neural networks are used as a front-end: the first one models rhythmic content on multiple frequency bands, while the second one models the harmonic content of the signal. The output activations are then combined and fed into a dynamic Bayesian network which acts as a rhythmical language model. We show on seven commonly used datasets of Western music that the system is able to achieve state-of-the-art results.

7.1 Introduction

The automatic analysis of the metrical structure in an audio piece is a long-standing, ongoing endeavour. A good underlying meter analysis system is fundamental for various tasks like automatic music segmentation, transcription, or applications such as automatic slicing in digital audio workstations.

The meter in music is organised in a hierarchy of pulses with integer related frequencies. In this work, we concentrate on one of the higher levels of the metrical hierarchy, the *measure* level. The first beat of a musical measure is called a *downbeat*, and this is typically where harmonic changes occur or specific rhythmic pattern begin (Klapuri et al., 2006).

The first system that automatically detected beats and downbeats was proposed by Goto and Muraoka (1999). It modelled three metrical levels, including the measure level by finding chord changes. Their system, built upon hand-designed features and rules, was reported to successfully track downbeats in $4/4$ music with drums. Since then, much has changed in the meter tracking literature. A general trend is to go from hand-crafted features and rules to automatically learned ones. In this line, rhythmic patterns are learned from data and used as observation model in probabilistic state-space models (Klapuri et al., 2006; Peeters and Papadopoulos, 2011; Krebs et al., 2013). Support Vector Machines (SVMs) were first applied to downbeat tracking in a semi-automatic setting (Jehan, 2005) and later used in a fully automatic system that operated on several beat-synchronous hand-crafted features (Durand et al., 2014). The latter system was later refined by using convolutional neural networks (ConvNets) instead of SVMs and a new set of features (Durand et al., 2015, 2016) and is the current state-of-the-art in downbeat tracking on Western music.

Recurrent neural networks (RNNs) are Neural Networks adapted to sequential data and therefore are the natural choice for sequence analysis tasks. In fact, they have shown success in various tasks such as speech recognition (Hannun et al., 2014), handwriting recognition (Graves et al., 2009) or beat tracking (Böck and Schedl, 2011). In this work, we would like to explore the application of RNNs to the downbeat tracking problem. We describe a system that detects downbeats from a beat-synchronous input feature sequence, analyse the performance of two different input features, and discuss shortcomings of the proposed model. We report state-of-the-art performance on seven datasets.

The paper is organised as follows: In Section 7.2 we describe the proposed RNN-based downbeat tracking system, in Section 7.3 we explain the experimental set-up of our evaluation and present and discuss the results in Section 7.4.

7.2 Method

An overview of the system is shown in Fig. 7.1. Two beat-synchronised feature streams (Section 7.2.1) are fed into two parallel RNNs (Section 7.2.2) to obtain a downbeat activation function which indicates the probability whether a beat is a downbeat. Finally, the activation function is decoded into a sequence of downbeat times by a dynamic Bayesian network (DBN) (Section 7.2.3).

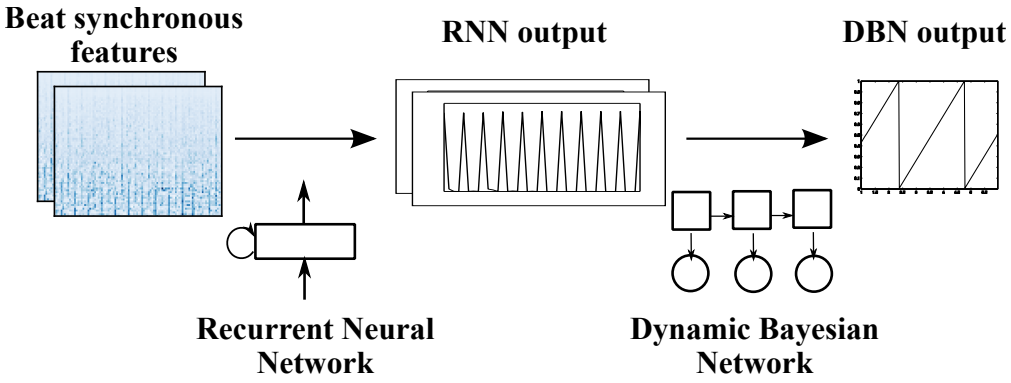


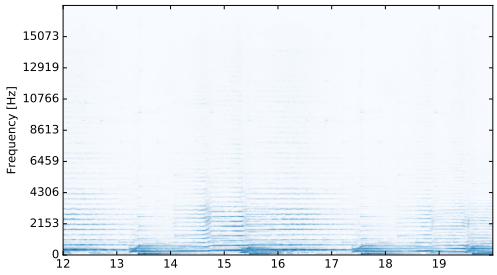
Figure 7.1: Model overview

7.2.1 Feature extraction

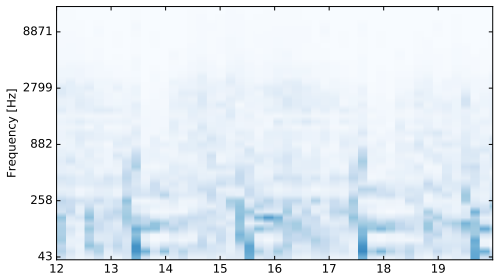
In this work we assume that the beat times of an audio signal are known, by using either hand-annotated or automatically generated labels. We believe that the segmentation into beats makes it much more easy for the subsequent stage to detect downbeats because it does not have to deal with tempo or expressive timing on one hand and it greatly reduces the computational complexity by both reducing the sequence length of an excerpt and the search space. Beat-synchronous features have successfully been used before for downbeat tracking (Davies and Plumbley, 2006; Papadopoulos and Peeters, 2011; Durand et al., 2015). Here, we use two features: A spectral flux with logarithmic frequency spacing to represent percussive content (*percussive feature*) and a chroma feature to represent the harmonic progressions throughout a song (*harmonic feature*).

Percussive feature

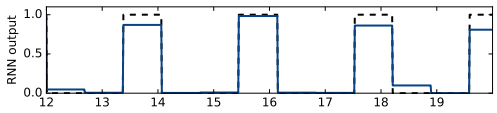
As a percussive feature, we compute a multi-band spectral flux: First, we compute the magnitude spectrogram by applying the Short-time Fourier Transform (STFT) with a Hann window, hopsize of 10ms, and a frame length of 2048 samples, as shown in Fig. 7.2a. Then, we apply a logarithmic filter bank with 6 bands per octave, covering the frequency range from 30 to 17 000 Hz, resulting in 45 bins in total. We compress the magnitude by applying the logarithm and finally compute for each frame the difference between the current and the previous frame. The feature sequence is then beat-synchronised by only keeping the mean value per frequency bin in a window of length Δ_b/n_p, where Δ_b is the beat period and $n_p = 4$ is the number of beat subdivisions, centred around the beginning of a beat subdivision. An example of the percussive feature is shown in Fig. 7.2b.



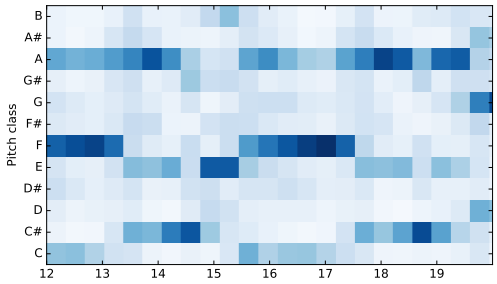
(a) Spectrogram



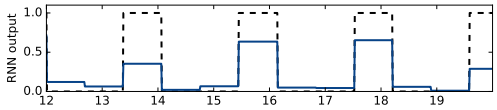
(b) Beat-synchronous percussive feature



(c) Activation and target of the rhythmic network



(d) Beat-synchronous chroma feature



(e) Activation and target of the harmonic network

Figure 7.2: Visualisation of the two feature streams and their corresponding network output of an 8-second excerpt of the song *Media-105701* (Ballroom dataset). The dashed line in (c) and (e) represents the target (downbeat) sequence, the solid line the networks' activations. The x-axis shows time in seconds. The time resolution is one fourth of the beat period in (b), and half a beat period in (d).

Harmonic feature

As harmonic feature, we use the *CLP* chroma feature (Müller and Ewert, 2011) with a frame rate of 100 frames per second. We synchronise the features to the beat by computing the mean over a window of length Δ_b/n_h, yielding $n_h = 2$ feature values per beat interval. We found that for the harmonic feature the resolution can be lower than for the percussive feature, as for chord changes the exact timing is less critical. An example of the harmonic feature is shown in Fig. 7.2d.

7.2.2 Recurrent Neural Network

RNNs are the natural choice for sequence modelling tasks but often difficult to train due to the exploding and vanishing gradient problems. In order to overcome these problems when dealing with long sequences, Long-Short-Term memory (LSTM) networks were proposed (Hochreiter and Schmidhuber, 1997). Later, Cho et al. (2014) proposed a simplified version of the LSTMs named Gated Recurrent Units (GRUs), which were shown to perform comparable to the traditional LSTM in a variety of tasks and have less parameters to train. Therefore, we will use GRUs in this paper.

The time unit modelled by the RNNs is the *beat period*, and all feature values that fall into one beat are condensed into one vector. E.g., using the percussive feature with 45 frequency bins and a resolution of $n_p = 4$ beat subdivisions yields an input dimension of $45 \times 4 = 180$ for the rhythmic RNN. In comparison to an RNN that models *subdivisions* of the beat period as underlying time unit, this vectorisation of the temporal context provided an important speed-up of the network training due to the reduced sequence length, while maintaining the same level of performance.

In preliminary tests, we investigated possible architectures for our task and compared their performances on the validation set (see Section 7.3.3). We made the following discoveries: First, adding bidirectional connections to the models was found to greatly improve the performance. Second, the use of LSTMs/GRUs further improved the performance compared to the standard RNN. Third, using more than two layers did not further improve the performance.

We therefore chose to use a two layer bidirectional network with GRU units and standard tanh non-linearity. Each hidden layer has 25 units. The output layer is a dense layer with one unit and a sigmoid non-linearity. Due to the different number of input units the rhythmic model has approximately 44k, and the harmonic model approximately 19k parameters.

The activations of both the rhythmic and harmonic model are finally averaged to yield the input activation for the subsequent DBN stage.

7.2.3 Dynamic Bayesian Network

The language model incorporates musical prior knowledge into the system. In our case it implements the following assumptions:

1. Beats are organised into bars, which consist of a constant number of beats.
2. The time signature of a piece determines the number of beats per bar.
3. Time signature changes are rare within a piece.

The DBN stage is similar to the one used by Durand et al. (2015), with three differences: First, we model beats as states instead of tatums. Second, as our data mainly contains $3/4$ and $4/4$ time signatures, we only model these two. Third, we force the state sequence to always transverse a whole bar from left to right, i.e., transitions from beat 2 to beat 1 are not allowed. In the following we give a short review of the DBN stage.

A state $s(b, r)$ in the DBN state space is determined by two hidden state variables: the beat counter b and the time signature r. The beat counter counts the beats within a bar $b \in \{1..N_r\}$ where N_r is the number of beats in time signature r. E.g., $r \in \{3, 4\}$ for the case where a 3/4 and a 4/4 time signature are modelled. The state transition probabilities can then be decomposed using

$$P(s_k|s_{k-1}) = P(b_k|b_{k-1}, r_{k-1}) \times P(r_k|r_{k-1}, b_k, b_{k-1}) \tag{7.1}$$

where

$$P(b_k|b_{k-1}, r_{k-1}) = \begin{cases} 1 & \text{if } b_k = (b_{k-1} \bmod r_{k-1}) + 1 \\ 0 & \text{otherwise.} \end{cases} \tag{7.2}$$

Eq. 7.2 ensures that the beat counter can only move steadily from left to right. Time signature changes are only allowed to happen at the beginning of a bar ($(b_k < b_{k-1})$), as implemented by

$$\begin{aligned} &\text{if} \quad (b_k < b_{k-1}) \\ &\qquad P(r_k|r_{k-1}, b_k, b_{k-1}) = \begin{cases} 1 - p_r & \text{if } (r_k = r_{k-1}) \\ p_r / R & \text{if } (r_k \neq r_{k-1}) \end{cases} \\ &\text{else} \\ &\qquad P(r_k|r_{k-1}, b_k, b_{k-1}) = 0 \end{aligned} \tag{7.3}$$

where p_r is the probability of a time signature change. We learned p_r on the validation set and found $p_r = 10^{-7}$ to be an overall good value, which makes time signature changes improbable but possible. However, the exact choice of this parameter is not critical, but it should be greater than zero as mentioned in Section 7.4.5.

As the sigmoid of the output layer of the RNN yields a value between 0 and 1, we can interpret its output as the probability that a specific beat is a downbeat and use

it as observation likelihood for the DBN. As the RNN outputs a posterior probability $P(s|\text{features})$, we need to scale it by a factor $\lambda(s)$ which is proportional to $1/P(s)$ in order to obtain

$$P(\text{features}|s) \propto P(s|\text{features})/P(s), \tag{7.4}$$

which is needed by the observation model of the DBN. Experiments have shown that a value of $\lambda(s(b = 1, r)) = 100$ for downbeat states and $\lambda(s(b > 1, r)) = 1$ for the other states performed best on our validation set, and will be used in this paper.

Finally, we use a uniform initial distribution over the states and decode the most probably state sequence with the Viterbi algorithm.

7.3 Experiments

System	Ballr.	Beatl.	Hain.	RWC	Rob.	Klap.	Rock	Mean
With annnotated beats:								
Rhythmic	83.9	87.1	75.7	91.9	93.4	-	87.0	84.4
Harmonic	77.2	89.9	80.1	92.9	92.6	-	86.0	82.2
Combined	91.8	89.6	83.6	94.4	96.6	-	89.4	90.4
With detected beats:								
Combined	80.3	79.8	71.3	82.7	83.4	69.3	79.0	77.3
Durand et al. (2016)	77.8	81.4	65.7	86.1	83.7	68.9	81.3	76.1
Beat tracking results:								
Krebs et al. (2015a)	89.0	88.4	88.2	88.6	88.2	85.2	90.5	88.3

Table 7.1: Mean downbeat tracking F-measures across all datasets. The last column shows the mean over all datasets used. The last row shows beat tracking F-measure scores of the beat tracking system (Böck et al., 2014; Krebs et al., 2015a).

7.3.1 Data

In this work, we restrict the data to Western music only and leave the evaluation of Non-Western music for future work. The following datasets are used:

Ballroom (Gouyon et al., 2006; Krebs et al., 2013): This dataset consists of 685 unique 30 second-long excerpts of Ballroom dance music. The total length is 5h 57m.

Beatles (Davies et al., 2009): This dataset consists of 180 songs of the Beatles. The total length is 8h 09m.

Hainsworth (Hainsworth and Macleod, 2004): This dataset consists of 222 excerpts, covering various genres. The total length is 3h 19m.

RWC Pop (Goto et al., 2002): This dataset consists of 100 American and Japanese Pop songs. The total length is 6h 47m.

Robbie Williams (Giorgi et al., 2013): 65 full songs of Robbie Williams. The total length is 4h 31m

Rock (De Clercq and Temperley, 2011): This dataset consists of 200 songs of the Rolling Stone magazine's list of the '500 Greatest Songs of All Time'. The total length is 12h 53m.

Klapuri (Klapuri et al., 2006): This dataset consists of 320 excerpts, covering various genres. The total length is 4h 54m. The beat annotations of this dataset have been made independently of the downbeat annotations and therefore do not always match. Hence, we cannot use the dataset in experiments that rely on annotated beats.

7.3.2 Evaluation measure

For the evaluation of downbeat tracking we follow Durand et al. (2015); Krebs et al. (2015a) and report the F-measure which is computed by $F = 2RP/(R + P)$, where the recall R is the ratio of correctly detected downbeats within a $\pm 70ms$ window and the total number of annotated downbeats, and the precision P is the ratio of correctly detected downbeats within this window and all the reported downbeats.

7.3.3 Training procedure

All experiments in this section have been carried out using the leave-one-dataset-out approach, to be as comparable as possible with the setting of Durand et al. (2016). After removing the test dataset, we use 75% of the remaining data for training and 25% for validation. To cope with the varying lengths of the audio excerpts, we split the training data into segments of 15 beats and an overlap of 10 beats. For training, we use cross entropy cost, and AdaGrad (Duchi et al., 2011) with a constant learn rate of 0.04 for the rhythmic model and 0.02 for the harmonic model. The hidden units and the biases are initialised with zero, and the weights of the network are randomly sampled from a normal distribution with zero mean and a standard deviation of 0.1. We stop the learning after 100 epochs or when the validation error does not decrease for 15 epochs. For training the GRUs, we used the Lasagne framework (Dieleman et al., 2015).

7.4 Results and Discussion

7.4.1 Influence of features

In this section we investigate the influence of the two different input features described in Section 7.2.1.

The performance of the two different networks is shown in the upper part of Table 7.1. Looking at the mean scores over all datasets, the rhythmic and harmonic network achieve a comparable performance. The biggest difference between the two was found in the *Ballroom* and the *Hainsworth* dataset, which we believe is mostly due to differing musical content. While the *Ballroom* set consists of music with clear and prominent rhythm which the percussive feature seems to capture well, the *Hainsworth* set also includes chorales with less clear-cut rhythm but more prominent harmonic content which in turn is better represented by the harmonic feature. Interestingly, combining both networks (by averaging the output activations) yields a score that is almost always higher than the score of the single networks. Apparently, the two networks concentrate on different, relevant aspects of the audio signal and combining them enables the system exploiting both. This is in line with the observations by Durand et al. (2016) who similarly combined the output of three networks in their system.

7.4.2 Estimated vs. annotated beat positions

In order to have a fully automatic downbeat tracking system we use the beat tracker proposed by Böck et al. (2014) with an enhanced state space (Krebs et al., 2015a) as a front-end to our system.[1] We show the beat tracking F-measures per dataset in the bottom row of Table 7.1. With regard to beat tracking, the datasets seem to be balanced in terms of difficulty.

The detected beats are then used to synchronise the features of the test set.[2] The downbeat scores obtained with the detected beats are shown in the middle part of Table 7.1. As can be seen, the values are around $10\% - 15\%$ lower than if annotated beats were used. This makes sense, since an error in the beat tracking stage cannot be corrected in a later stage. This might be a drawback of the proposed system compared to Durand et al. (2016), where the tatum (instead of the beat) is the basic time unit and the downbeat tracking stage can still decide whether a beat consists of one, two or more tatums.

Although the beat tracking performance is balanced among the datasets, we find clear differences in the downbeat tracking performance. For example, while the beat tracking performance on the *Hainsworth* and the *Robbie Williams* dataset are similar, the downbeat accuracy differs more than 12%. Apparently, the mix of genres, including

[1] We use the *DBNBeatTracker* included in *madmom* (Böck et al., 2016a) version 0.13.

[2] We took care that there is no overlap between the train and test sets.

time signatures of $2/2$, $3/2$, $3/4$ and $6/8$, in the *Hainsworth* set represents a challenge to downbeat tracking compared to the more simple *Robbie Williams*, which mostly contains $4/4$ time signatures.

7.4.3 Importance of the DBN stage

System	annotated	detected
RNN	85.0	73.7
RNN+DBN	90.4	77.3

Table 7.2: Mean downbeat tracking F-measures across all datasets of the proposed, combined system. *annotated* and *detected* means that annotated or detected beats were respectively used to synchronise the features. RNN uses peak-picking to select the downbeats, while RNN+DBN uses the DBN language model.

To assess the importance of the DBN stage (Section 7.2.3) we implemented a simple baseline, which simply reports downbeats if the resulting (combined) RNN activations exceed a threshold. A threshold of 0.2 was found to yield the best results on the validation set. In Table 7.2, we show the results of the baseline (RNN) and the results of the combined system (RNN+DBN). As can be seen, the combination of RNN and DBN significantly outperforms the baseline, confirmed by a *Wilcoxon signed-rank* test with $p < 0.01$.

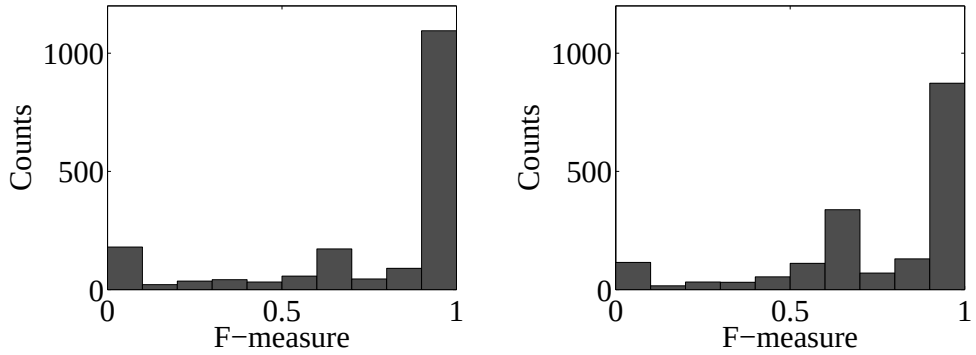
7.4.4 Comparison to the state-of-the-art

In this section we investigate the performance of our system in relation to the state-of-the-art in downbeat tracking, represented by Durand et al. (2016). Unfortunately, a direct comparison is hindered by various reasons: The datasets used for training the ConvNets (Durand et al., 2016) are not freely available and the beat tracker at their input stage is different to the one that we use in this work. Therefore, we can only check whether the whole end-to-end system is competitive and leave a modular comparison of the approaches to future work.

In the middle of Table 7.1 we show the results of the system described by Durand et al. (2016), as provided by the authors. The last column shows the mean accuracy over all 1771 excerpts in our dataset. A *paired-sample t-test* did not show any statistically significant differences in the *mean* performance between the two approaches considering all data points. However, a *Wilcoxon signed-rank* test revealed that there is a significant ($p < 0.01$) difference in the *median* F-measure over all data points, which is 89.7% for Durand et al. (2016) and 96.2% for the proposed system. Looking at histograms of the obtained scores (see Fig. 7.3), we found a clear peak at around

66% F-measure, which is typically caused by the beat tracking stage reporting half or double of the correct tempo. The peak is more prominent for the system by Durand et al. (2016) (Fig. 7.3b), hence we believe the system might benefit from a more accurate beat tracker.

From this we conclude that overall the proposed system (in combination with the beat tracker proposed by Böck et al. (2014); Krebs et al. (2015a)) performs comparable to the state of the art when looking at the mean performance and even outperforms the state of the art in terms of the median performance.



(a) Proposed system (b) Reference system

Figure 7.3: Histogram of the downbeat F-measures of the proposed system (a) and the reference system (Durand et al., 2016) (b)

7.4.5 Error analysis

In order to uncover the shortcomings of the proposed model we analysed the errors of a randomly-chosen, small subset of 30 excerpts. We identified two main factors that lead to a low downbeat score. The first one, obviously, are beat tracking errors which get propagated through to the downbeat stage. Most beat tracking errors are octave errors, and among them, the beat tracker mostly tapped twice as fast as the groundtruth tempo. In some cases this is acceptable and therefore would make sense to also allow these metrical levels as, e.g., done by Klapuri et al. (2006). The second common error is that the downbeat tracker chooses the wrong time signature or has problems following time signature changes or coping with inserted or removed beats. Phase errors are relatively rare. Changing time signatures appear most frequently in the *Beatles* dataset. For this dataset, reducing the transition probability of time signature changes p_r from 10^{-7} to 0 leads to a relative performance drop of 6%, while the results for other datasets remain largely unaffected. Besides, the used datasets mainly contain $3/4$ and $4/4$ time signatures making it impossible for the RNN to learn some-

thing meaningful about other time signatures. Here, creating a more balanced training set regarding time signatures would surely help.

7.5 Conclusions and future work

We have proposed a downbeat tracking back-end system that uses recurrent Neural networks (RNNs) to analyse a beat-synchronous feature stream. With estimated beats as input, the system performs comparable to the state of the art, yielding a mean downbeat F-measure of 77.3% on a set of 1771 excerpts of Western music. With manually annotated beats the score goes up to 90.4%.

For future work, a good modular comparison of downbeat tracking approaches needs to be undertaken, possibly with collaboration between several researchers. In particular, standardised dataset train/test splits need to be defined. Second, we would like to train and test the model with non-Western music and 'odd' time signatures, such as done by Holzapfel et al. (2014).

The source code will be released as part of the *madmom* library (Böck et al., 2016a), including all trained models and can be found together with additional material under `http://www.cp.jku.at/people/krebs/ismir2016/index.html`.

Chapter 8

Online Beat and Downbeat Tracking

All the algorithms presented so far have been developed and evaluated for the offline case, where a whole piece of music is fed into the system and the system outputs a metrical analysis of the entire piece at once. In this chapter we will modify the system to work online, meaning that the system processes an audio signal frame-by-frame and analyses each frame using only information from the current and past frames. In this case, it is not possible to correct a decision once it is made. Such a system can be used for many applications, e.g., a synthetic drummer who listens to a musician and, based on his analysis, accompanies him/her, or an application that shows the tempo of a music piece while you listen to it. Two examples for such applications are given in Chapter 9.

8.1 System description

In the following we will describe the steps that are necessary to turn the models proposed so far into online-capable meter analysis systems. Both acoustic and language model (see Fig. 1.1) have to work causally, i.e., it is not allowed to use future information to make a decision about the current time. The GMM-based acoustic models (Chapter 3-6) already satisfy this condition. In the case of RNN-based acoustic models (Chapter 6 and 7) we have to use *unidirectional* RNNs instead of the *bidirectional* ones. To obtain a causal language model, we simply have to replace the Viterbi algorithm by the Forward algorithm when doing inference. In the following, we additionally refine the language model to explicitly detect non-metric content and then investigate how to select a hidden state from the filtering distribution (Eq. 2.2) at each time step.

8.1.1 Dealing with non-metric content

So far, we have assumed that an audio clip *only* consists of metrical content, and thus will try to detect beats and downbeats, even if there is silence. To avoid these false positive detections, we add a 'silence state' to our model. This state can be entered from each tempo state at the end of a beat period with probability $p_{to\_silence}$. From this state, there are transitions to the beginning of a beat in all tempi with probability $p_{from\_silence}$. It also has a high self-transition probability. To obtain the observation probabilities $P(y|x_s)$ (observing y being in the silence state x_s) we can either learn this distribution from data, or we simply re-use the observation probability of a non-beat position in the state space. For convenience, we adapted the latter approach.

8.1.2 Inference

So far, we have computed the MAP-sequence either by Viterbi decoding in the discrete state space (see Chapters 3-7) or by selecting the particle with highest weight at the end of a piece in the mixed discrete-continuous state space (Chapter 5). For an online system, we will need to compute the filtering distribution $P(\mathbf{x}_k|\mathbf{y}_{1:k})$ (see Section 2.2) for each time frame k which is the probability distribution over all hidden states $\mathbf{x}_k$ given all observations which have happened up to the current frame $\mathbf{y}_{1:k}$. Once we have computed the filtering distribution, we have to select one hidden state to finally report it. We will call this 'winning' state, which will be reported at each frame, the *win-state* $\hat{x}$. In the following we will present three methods how to compute $\hat{x}$:

Max observation likelihood

The first and simplest method is to bypass the DBN and to use the observation probability directly to select the win-state by

$$\hat{x}_t^{obs} = \arg\max_{x_t} P(\mathbf{y}_k|\mathbf{x}_k). \tag{8.1}$$

In this case the performance will depend highly on the quality of the observation model. If complex observation models are used (e.g., RNNs as proposed by Böck and Schedl (2011)) this method could work sufficiently well.

Max filtering

The second method is to simply report the state with the highest filtering probability by computing

$$\hat{x}_t^{max} = \arg\max_{x_t} P(\mathbf{x}_k|\mathbf{y}_{1:k}). \tag{8.2}$$

This method will work well if $P(\mathbf{x}_k|\mathbf{y}_{1:k})$ has *one* clear peak. For multi-modal distributions selecting the maximum at each time frame can yield states which jump from one mode to another.

Constrained max filtering

Finally, the third method tries to prevent the states from jumping by enforcing continuity between adjacent win-states. It does so by using another DBN, which uses the filtering distribution as 'observation model', and then select the win-state as the most probably state given the last hidden state by

$$\hat{x}_t^{dbn} = \arg\max_{x_t}[P(\mathbf{x}_k|\mathbf{y}_{1:k}) * Q(\mathbf{x}_k|\hat{x}_{k-1})], \tag{8.3}$$

where $Q(\mathbf{x}_k|\hat{x}_{k-1})$ are the probabilities of going from the previous win-state $\hat{x}_{k-1}$ to $\mathbf{x}_k$. We first set $Q(x_k|x_{k-1}) = P(x_k|x_{k-1})$ but achieved better results with an extended transition model: $P(x_k|\hat{x}_{k-1})$ defines a set of possible target states $\tilde{x_k}$, which can be reached from $\hat{x}_{k-1}$ with a non-zero transition probability. We now add additional transitions from $\hat{x}_{k-1}$ to all neighbouring states of $\tilde{x_k}$ (states that sit left, right, above and below in the state grid) to the transition model. The purpose of this extended transition model is to enlarge the space where the win-state is chosen from. However, this method has the problem that the win-state can be stuck with one mode of the distribution, even if globally this is not the best solution. To prevent this case, we reset the win-state to the state with the global maximum filtering probability, whenever the probability of the current win-state is below a certain threshold. This threshold is defined as a fraction of the current maximum filtering probability. I.e., as soon as $P(\hat{x}_t^{dbn}|\mathbf{y}_{1:k}) < \max P(\mathbf{x}_k|\mathbf{y}_{1:k})/f$, we set the win-state to the state with the maximum filtering probability.

8.2 Evaluation

In the following, we will compare the performance of the online tracking model with the offline variant, which uses Viterbi decoding. For the online systems, we compare the three methods of choosing the win-state presented in Section 8.1.2. The methods will be compared using the *RNN-BeatTracker* system described in Chapter 6, but are applicable to all systems of this thesis.

Generally, there are two challenges for online meter estimation: First, gradual tempo fluctuations are difficult to recognize because information from the future is not available, and second, sudden transitions (e.g., between songs) cause confusion as the system needs time adapt to a new tempo and phase of beats/downbeats. To test the system under these two conditions, we generated a special dataset for each case.

We first chose the parameters to maximise the performance on a training set consisting of a modified Beatles and Rock dataset (Section 2.4). Then we give results on the (modified) Ballroom, and Hainsworth in Table 8.1 and 8.3. As the SMC dataset already contains tempo variations, we did not modify it there.

8.2.1 Evaluation metrics

For a definition of *FM*, *CMLt* and *AMLt*, please see Sections 4.4.2 and 5.5.2. *CMLc* is the ration of the duration of the longest correctly detected segment and the length of a song. In *AMLc*, double and half of the annotated tempo are also counted as correct, as well as tapping on the offbeat.

8.2.2 Observation model

For the evaluation, we will use RNNs similar to the ones proposed by Böck and Schedl (2011). To make them online-capable, we use uni-directional RNNs, and the input features are computed with a (single) frame length of 2048 bins and 6 bands per octave. The RNNs were trained by Sebastian Böck.

8.2.3 Data modifications

Time-varying tempo

One challenge for an online beat tracking system is to follow a time-varying tempo trajectory. As most datasets mostly contain excerpts with a more or less constant tempo, we created new datasets: We took existing datasets and introduced a time-varying tempo curve by time-stretching the audio (and the annotations) using the *rubberband* library (Cannam, 2012).

The sequence of modified beat periods $\{\Delta'(b) : b = 1, 2, 3, ...\}$) is defined by two variables: First, the cycle length of the tempo variations in beats B, which is computed by

$$B = \text{cycle\_length [min]} * \text{tempo [BPM]}, \tag{8.4}$$

where we use a cycle length of one minute and *tempo* is the median tempo of a song. The higher B, the slower are the induced tempo variations. Second, the extent of tempo variation as fraction (Δ_{min} and Δ_{max}) of the original tempo. Then, we can modify the original sequence of beat periods $\{\Delta(b)\}$ using

$$\Delta'(b) = \Delta(b) * (\sin(2 * \pi * \frac{b}{B}) + 1) * \frac{\Delta_{max} - \Delta_{min}}{2} + \Delta_{min} \tag{8.5}$$

where $\Delta(b)$ is the original inter-beat interval between beat b and $b - 1$, B is the cycle length in beats and Δ_{min}, Δ_{max} are the minimum and maximum stretching factors. For

our experiments, we chose $\Delta_{min} = 0.7$ and $\Delta_{max} = 1.3$. An example of a modified tempo curve is shown in Figure 8.1, together with the original tempo curve.

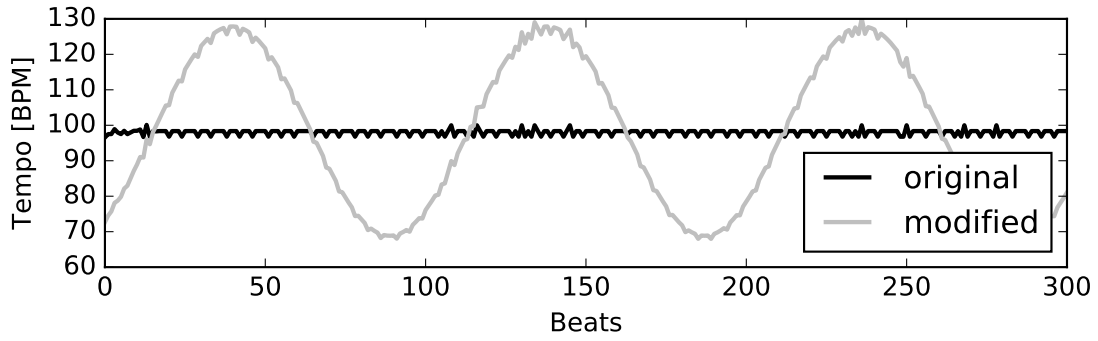


Figure 8.1: Original and modified tempo curve of the song *Feeling In My Heart* of the RWC dataset. Here, $B = 100$, $\Delta_{min} = 0.7$ and $\Delta_{max} = 1.3$

Streaming dataset

To test the reaction time of the system to changing inputs, we created two streaming datasets, similar to Collins (2006); Oliveira et al. (2012). The dataset consists of 200 data streams, which again consist of 5 different concatenated excerpts, each 15 seconds long, separated by 3 seconds of silence. The silence is important for the online use case, as we do not want the tracker to report any meter events in pauses between songs. The 15-seconds-long excerpts were randomly chosen from any position within a piece. We created two datasets, one for parameter tuning and one for testing.

8.2.4 Results and Discussion

For the experiments, we use the best performing parameter values on the training set. These are a tempo transition coefficient $\lambda = 100$ (Section 6.2.3), silence transition probabilities $p_{from\_silence} = 1e^{-4}$, $p_{to\_silence} = 1e^{-5}$, and a reset threshold for the constrained max filtering $f = 1.5$.

Time-varying tempo

The beat tracking results on datasets with varying tempo are shown in Table 8.1. As expected, modifying the tempo did generally decrease the performance of the systems, comparing with the scores on the original datasets (Table 8.2). This is clearest when looking at the continuity-based metrics.

	FM	CMLc	CMLt	AMLc	AMLt
Ballroom-tempo					
Max observation likelihood	83.5	40.3	60.0	43.1	64.8
Max filtering	80.4	49.1	61.3	59.1	73.6
Constrained max filtering	**84.0**	**51.9**	**63.5**	**60.8**	**74.7**
Viterbi	86.5	69.4	71.5	87.0	89.8
Hainsworth-tempo					
Max observation likelihood	73.2	27.7	50.1	31.3	56.6
Max filtering	76.7	**45.4**	**60.9**	**50.9**	**68.5**
Constrained max filtering	**80.0**	44.7	60.1	50.8	67.9
Viterbi	83.3	68.1	72.3	79.6	84.3
SMC					
Max observation likelihood	28.9	6.2	11.8	8.6	16.0
Max filtering	39.9	**17.2**	**28.3**	**22.1**	**36.7**
Constrained max filtering	**40.7**	13.4	24.2	17.6	30.6
Viterbi	44.3	27.0	37.5	37.2	53.9

Table 8.1: Beat tracking accuracies on the tempo modified datasets.

The SMC set seems to be the most difficult one, as the scores are drastically lower than the other sets. This set contains expressive tempo curves which are apparently less predictable than the tempo modulation that we have introduced.

Obviously, the offline version (using Viterbi) performs best in all metrics over all datasets. Comparing the online systems, we find that the max-filtering approach performs best in most situations. Adding constraints on the possible target states (Constrained max filtering) did not improve the performance in the majority of cases.

Streaming

The results on the streaming dataset are shown in Table 8.3. Obviously, the CMLc and AMLc values are lower than on other datasets. Interestingly, the simple max-observation-likelihood approach performs very well here. This can be explained by the fact that in the streaming scenario, continuity is very difficult to achieve for an online system. Therefore, the impact of the post-processing DBN, which smooths the beat trajectory, is lower. It is more important to follow fast changes in the music, and the max-observation-likelihood approach has the lowest reaction time.

	FM	CMLc	CMLt	AMLc	AMLt
Ballroom					
Max observation likelihood	84.8	50.8	67.6	53.1	71.8
Max filtering	**86.6**	**62.6**	**73.1**	**67.7**	**79.8**
Constrained max filtering	86.5	61.7	72.8	66.6	79.5
Viterbi	89.5	77.7	81.9	85.9	90.7
Hainsworth					
Max observation likelihood	72.4	38.1	54.6	44.0	64.5
Max filtering	**78.1**	**52.9**	**62.0**	**63.1**	**76.0**
Constrained max filtering	78.0	52.2	61.9	61.9	75.7
Viterbi	82.4	67.2	71.8	83.8	90.0

Table 8.2: Beat tracking accuracies on the original (not tempo-modified) datasets.

	FM	CMLc	CMLt	AMLc	AMLt
Ballroom-stream					
Max observation likelihood	**83.2**	**21.7**	64.1	**21.9**	64.4
Max filtering	82.7	20.9	**64.4**	21.2	**64.5**
Constrained max filtering	82.4	20.8	64.1	21.0	64.2
Viterbi	87.7	40.6	74.2	40.7	74.2

Table 8.3: Beat tracking accuracies on the streaming dataset.

8.3 Conclusions

Our meter analysis system can easily be turned into an online-capable system by replacing the Viterbi decoding by the forward path. We compared three methods to select a 'winning' state which is then reported to the user online. We find that selecting the most probable state based on the filtering distribution leads in average to the best performance in both conditions of changing tempo and concatenated excerpts (streaming).

Chapter 9

Applications, Code, and Data

In this chapter, I present applications, code and a dataset that have been developed during the time of writing this thesis.

9.1 Applications

9.1.1 The automatic ballroom instructor

The automatic ballroom instructor is a system that teaches people how to dance to the (ballroom) music. With the technology presented in this thesis it is possible to obtain a metrical analysis of an audio clip. This metrical analysis can then be used to synchronise a video of dance moves to any given music piece. The synchronisation can either be done offline (in order to study dance moves at home), or online (assisting while you are dancing in a concert), similar to the system of Eyben et al. (2007). The system described in Chapter 3 can discriminate between eight standard ballroom dances and can therefore be used to automatically select the right dance moves.

An offline variant of such a system has been implemented by Martin Ennemoser as part of a student's project. Dance moves are predefined by storing the coordinates of the feet at each metrical position in configuration files. The user can then load an audio file, run the meter analysis and finally watch a video of the synchronised dance videos along with the music. A screenshot of the application is shown in Figure 9.1.

9.1.2 Drumotron 3000

The Drumotron 3000 is a real-time drum accompaniment system, which is able to listen to a live-musician, analyse the meter, and accompany the musician on an acoustic drum set (see Figure 9.4a). The main components are depicted schematically in Figure 9.2.

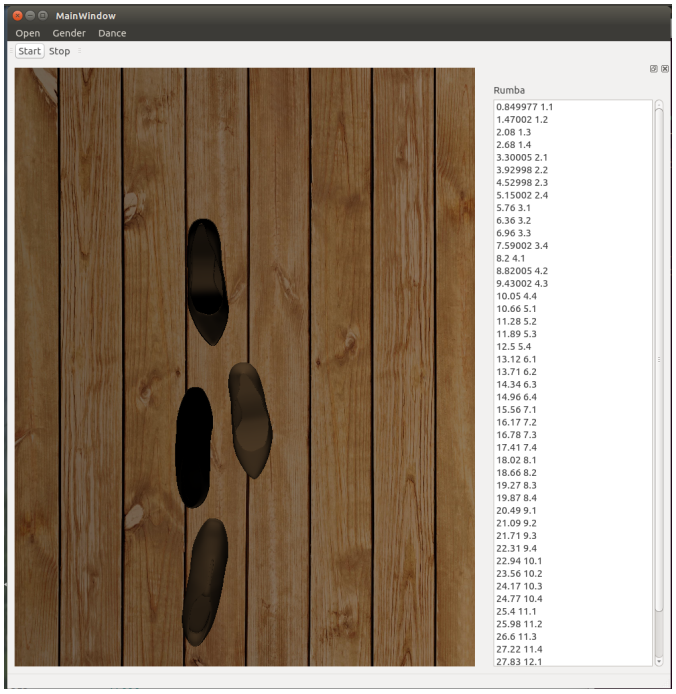


Figure 9.1: The automatic ballroom dance instructor

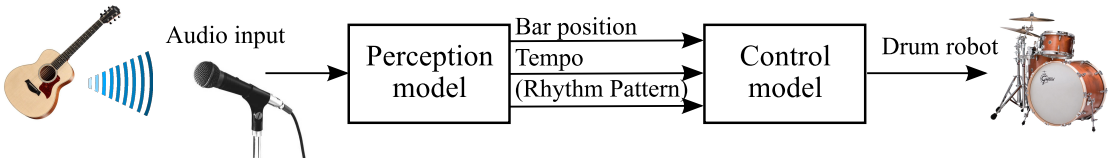


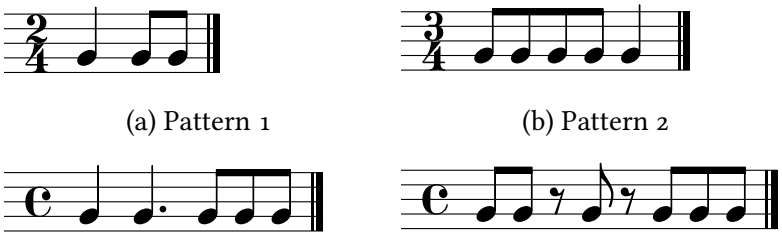
Figure 9.2: Drumotron 3000 system overview

A musician plays an instrument (usually guitar or piano). The sound is converted to a digital signal by a microphone and a A-D converter within an audio interface. It enters the *perception model*, where features are extracted, the observation likelihood is computed and finally metrical parameters such as tempo and bar position are inferred. These musical attributes are then converted into instructions for the drum robot: Given the bar position and a predefined drum pattern, it can be determined whether and which drum should be hit at a certain time. If the case, the command to hit the drum is sent to the control model. The control then sends signals to the servos which finally beat the specified drum.

Perception Model

The main task of the perception model is to analyse the meter of the audio input. It takes an audio signal as input, and outputs metrical parameters such as tempo, bar po-

sition, and rhythmic patterns. Since the beginning of writing this thesis, two versions (*D2015* and *D2016*) of the perception model have been developed.



(a) Pattern 1 (b) Pattern 2

(c) Pattern 3 (d) Pattern 4

Figure 9.3: Patterns of the *D2015* perception model.

Software *D2015* was developed in 2015 and is mainly based on the work presented in Chapter 3 and 6. I use GMMs as observation model and model bars, tempo and rhythmic patterns, using the improved state space of Chapter 6. *D2015* was mainly used in presentations for students, where I modelled the four rhythmic patterns shown in Fig. 9.3. For inference, the *constrained-max-filtering* approach (see Chapter 8) was found to be the best performing and therefore has been used. *D2015* is available in MATLAB for prototyping as well as in C++ for live demonstrations.

D2016 is being developed while writing this thesis in 2016. It uses the RNN-based observation model of Böck et al. (2016b) and models bars and tempo (no rhythmic patterns). As outlined in Chapter 8, the max-filtering inference approach was found to work best with this configuration.

Hardware *D2015* runs on a MacBook Pro with i5 Dual-Core @2.4 GHz. *D2016* is expected to run on a Raspberry Pi 3 Model B with a quad-core CPU @1.2 GHz.

Control model

The control model processes the metrical parameters output by the perception model, and sends commands to the drum robot whether and which drum to beat.

Software The control model of version *D2015* was developed by Harald Frostel and Rainer Kelz in C++ and Python. For each rhythmic pattern of the system, a drum pattern has been defined, which determines which drum has to be beaten at which bar position. Currently, Sebastian Böck and I are working on version *D2016* of the control model, which will run in pure Python/Cython.

Hardware An Arduino microcontroller is connected to the MacBook / Raspberry Pi via USB, and redirects the control signals to the servos which are attached to the single drums of the drum set (see Figure 9.4b).

(a) Musician jamming with Drumotron 3000 (b) Robot's snare arm

Figure 9.4: Drumotron 3000

Drum robot hardware

The hardware of the drum robot has been built by Rainer Kelz and Martin Preinfalk. It features four actuators, three driven by Modelcraft RS-2 Servos (for hi-hat, snare and one tom-tom drum), and one driven by a high power linear solenoid for the bass drum.

Practical issues

During the time of this thesis I did several demonstrations of the drumotron 3000 system version *D2015* accompanying a guitar player. Here are some subjective comments on the performance in practice. I found that following the player worked very well, even if the musician smoothly changes the tempo. Silence detection also worked very well. The main limiting factor was the dependency on the trained rhythmic patterns. Every time the musician played against the predefined patterns of the model, it would fail tracking the meter. But adding more and more patterns to the model increases the computational cost at one hand and leads to confusions between patterns on the other hands. This is also the main reason for developing version *D2016* of the system. *D2016* employs RNNs as observation model and does not explicitly model rhythmic patterns any more. I expect the new version to generalise much better over a variety of music styles.

9.2 Code

Most of the code which was developed for this thesis is published as part of two software toolboxes

BayesBeat is a MATLAB package for inferring metrical structure from musical audio using probabilistic models. It contains most of the material presented in Chapters 3, 5, and 6, and was used by Holzapfel et al. (2014); Srinivasamurthy et al. (2015); Holzapfel and Benetos (2016); Srinivasamurthy et al. (2016); Holzapfel and Grill (2016). It is written in MATLAB/C++ and available for download[1].

madmom is a PYTHON module for audio signal processing (Böck et al., 2016a). It contains most of the material presented in Chapters 3, 6, 7 as well as other works I contributed to, but which are not part of this thesis (Böck et al., 2012b,a; Böck et al., 2014, 2015, 2016b). It is written in Python/Cython and available for download[2].

9.3 Data

9.3.1 The Ballroom Dataset

Th Ballroom dataset was originally set up by Gouyon et al. (2004, 2006) for the purpose of genre classification and tempo estimation. The authors downloaded 698 publicly-available music excerpts, each around 30 seconds long, including annotations of tempo and dance style.[3]. Later, Dixon et al. (2004) annotated the first bar of each excerpt for their work on genre classification. For this thesis, I added beat and downbeat annotations, which are publicly available for download.[4] I use a version control system to be able to correct the annotations once new errors are identified. For the works in this thesis, I use version 1.2 of the annotations.

Sturm (2014) discovered that the original dataset contained 13 replicas. Therefore, I removed the replicated pieces and use only the remaining 685 unique excerpts in this thesis.

Audio quality According to Gouyon et al. (2004), 'the audio quality of this data is quite low, it was originally fetched in real audio format, with a compression factor of almost 22 with respect to the common 44.1 kHz 16 bits mono WAV format.'

[1] https://github.com/flokadillo/bayesbeat

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

[3] The data was originally fetched from http://www.ballroomdancers.com/Music/style.as. Now the dataset can be found under Gouyon (2006)

[4] https://github.com/CPJKU/BallroomAnnotations

Meter annotation strategy It is a well known problem that annotators disagree on the metrical level of a musical piece. For creating the meter annotations, I relied on the tempo restrictions that are given by the dance style label of the ballroom dances. I therefore chose the metrical level of the beats according to the provided tempo annotations. Note that this might cover only one part of the truth. Intros that do not contain any musical content were skipped. Generally, the dataset was annotated using two stages: First, I used the beat tracker of Böck and Schedl (2011) to interpolate the (manual) bar annotations kindly provided by Dixon et al. (2004). Then, I went through the dataset several times and manually corrected the beat and downbeat times.

Dance style	**# excerpts**	**tempo mean**	**tempo std**
Cha-Cha-Cha	107	122.6	2.12
Jive	60	165.9	2.36
Quickstep	81	204.0	3.16
Rumba	95	99.0	2.52
Samba	83	100.0	1.99
Tango	86	127.3	3.28
Viennese	65	177.6	3.31
Waltz	108	85.9	4.23
All	685	130.0	2.89
RWC classical	61	112.9	16.91
SMC	217	78.0	12.98
Rock	200	115.9	5.93
Carnatic	176	120.7	5.84
Beatles	179	117.4	4.80
Hainsworth	222	114.9	4.57
GTZAN	999	119.6	3.79
Ballroom	*685*	*130.0*	*2.89*
RWC pop	100	113.7	2.35
HJDB	235	152.2	2.07

Table 9.1: Tempo statistics

Dataset statistics The number of excerpts, tempo mean and tempo standard deviation for each dance style of the ballroom dataset as well as other datasets are shown in Table 9.1. The tempo mean is determined by computing the median tempo of each song and then computing the mean over the medians. For the tempo standard deviation, I

first 'normalise' each song to a median tempo of 120 BPM, compute the standard deviation for each song, and then compute the mean over the standard deviations. As can be seen from Table 9.1, Waltz, Tango and Quickstep have the most expressive tempo, as measured by the tempo standard deviation. But, in comparison to other datasets (see bottom of Table 9.1), this expressiveness is low. In fact, only the HJDB and the RWC pop dataset have less tempo variability. From this we can tell that the dataset is rather easy for automatic beat tracking, due to small amount of tempo variation. In this aspect, the SMC and RWC classical datasets pose a more difficult and interesting challenge.

Chapter 10

Conclusions

In this thesis, several methods for analysing the meter of music have been presented. Below, I summarise the key findings, and give ideas for future work.

Feature learning is a must. Recent advances in machine learning have shown the superiority of automatic feature learning over hand designed features. In the yearly MIREX [1] audio beat tracking evaluation, approaches using automatic feature learning outperform hand-designed approaches by a margin of 10% on average. For example, the best F-measure score on the MCK dataset is 63.9 for *SB9.2016* (the best score by a machine learning approach) and 53.7 for the *JZ2.2014* (the best score of a 'hand-designed' approach so far). This is also valid for more complex tasks such as downbeat tracking with usually more then 20% difference in F-measure[2].

Use common data and splits for evaluation. For future developments in the field of automatic meter analysis, it is indispensable to have common, open datasets with predefined train/test splits to be able to compare systems and system components in a rigorous way. For example, downbeat tracking systems using RNNs, CNNs, and GMMs have been proposed, but so far no fair comparison has been undertaken. Such a comparison will need a pre-defined training/validation/test data protocol with public data and a fixed low-level data representation in the style of the *ImageNetchallenge* (Russakovsky et al., 2015).

Guide the training process. During the evaluation of the systems I found that simple rules like 'downbeats are more likely when the harmony changes' or 'time signature changes occur very rarely' are not always captured by the learning method. This

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

[2] http://www.music-ir.org/mirex/wiki/2016:Audio_Downbeat_Estimation_Results

is also the reason that the DBN post-processing still has such a big impact on the results (see for example Table 7.2). The ultimate goal is to have strong feature learners that can solve a task in an end-to-end manner without any hand-designed intermediate stages, as it already has been pursued for speech recognition (Hannun et al., 2014). One important prerequisite to achieve this is a representative training corpus. This corpus should be tailored to contain enough stimuli for the feature learner to learn certain rules. For example, if I want the RNN to learn that downbeats occur at harmony changes, I could, for example, design a dataset where harmonic change is the only cue to recognize downbeats. Hopefully, this would force the learner to take it into account. Or, one could apply multi-task learning (Caruana, 1997) and guide the training process by solving two tasks at the same time. For example, joint training of chords and downbeats detection could make it easier for the learner to capture the relationship between harmony and downbeats. Another option is to design a feature (by hand) that represents harmony and nothing else (e.g., normalised chroma vectors) and feed this single feature to the learner (Papadopoulos and Peeters, 2011; Durand et al., 2015; Krebs et al., 2016). Due to the restricted input, the system is forced to learn the relation between harmony and meter.

Revisit the evaluation methods. Meter analysis is highly subjective, as well as potentially ambiguous (Stobart and Cross, 2000). Often, it is hard to decide whether a music piece is in $2/4$ or $4/4$. Sometimes, even both of them occur together at the same time, e.g., the guitar plays in $4/4$, while the drums play in $2/4$. In these cases, both annotations and evaluation metrics have to take into account this ambiguity and subjectivity. In the long run it is unavoidable to have multiple annotations of a piece, covering multiple annotators as well as multiple annotations per annotator. In addition, large-scale listening tests should be performed in order to test the perceptual relevance of currently used evaluation metrics, in the line of Davies and Böck (2014).

References

A. Allahverdyan and A. Galstyan. Comparative analysis of viterbi training and maximum likelihood estimation for HMMs. In *Advances in Neural Information Processing Systems*, page 1674–1682, 2011.

M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, 2002.

J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1047, 2005. ISSN 1063-6676.

J. A. Bilmes. Timing is of the essence: Perceptual and computational techniques for representing, learning, and reproducing expressive timing in percussive rhythm. Master's thesis, Massachusetts Institute of Technology, 1993.

S. Böck and M. Schedl. Enhanced beat tracking with context-aware neural networks. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2011.

S. Böck, A. Arzt, F. Krebs, and M. Schedl. Online real-time onset detection with recurrent neural networks. In *Proc. of the 15th International Conference on Digital Audio Effects*, 2012a.

S. Böck, F. Krebs, and M. Schedl. Evaluating the online capabilities of onset detection methods. In *Proceedings of the 14th International Conference on Music Information Retrieval (ISMIR)*, Porto, 2012b.

S. Böck, F. Krebs, and G. Widmer. A multi-model approach to beat tracking considering heterogeneous music styles. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, 2014.

S. Böck, F. Krebs, and G. 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)*, Malaga, 2015.

S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer. madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, 10 2016a. doi: 10.1145/2964284.2973795.

S. Böck, F. Krebs, and G. Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York, 2016b.

E. Bozdag. Bias in algorithmic filtering and personalization. *Ethics and Information Technology*, 15(3):209–227, 2013.

D. J. Cameron, J. Bentley, and J. A. Grahn. Cross-cultural influences on rhythm processing: Reproduction, discrimination, and beat tapping. *Frontiers in psychology*, 6, 2015.

C. Cannam. Rubber band library. `http://www.breakfastquay.com/rubberband/`, 2012. Online; Accessed: 2016-09-20.

R. Caruana. *Multitask Learning*. PhD thesis, Carnegie Mellon University Pittsburgh, 1997.

A. T. Cemgil and B. Kappen. Monte carlo methods for tempo tracking and rhythm quantization. *Journal of Artificial Intelligence Research*, 18(1):45–81, 2003.

A. T. Cemgil, B. Kappen, P. Desain, and H. Honing. On tempo tracking: Tempogram representation and kalman filtering. *Journal of New Music Research*, 29(4):259–273, 2000.

A. T. Cemgil, H. J. Kappen, and D. Barber. A generative model for music transcription. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):679–694, 2006.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv:1406.1078*, 2014.

N. Collins. Towards a style-specific basis for computational beat tracking. In *Proceedings of the 9th International Conference on Music Perception and Cognition (ICMPC)*, pages 461–467, Bologna, Italy, 2006.

T. Collins, S. Böck, F. Krebs, and G. Widmer. Bridging the audio-symbolic gap: The discovery of repeated note content directly from polyphonic music audio. In *AES 53rd International Conference on Semantic Audio*, London, 2014. Audio Engineering Society.

M. E. P. Davies and S. Böck. Evaluating the evaluation measures for beat tracking. In *Proceedings of the 15th International Conference on Music Information Retrieval (ISMIR)*, Taipei, 2014.

M. E. P. Davies and M. D. Plumbley. A spectral difference approach to downbeat extraction in musical audio. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Florence, 2006.

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

M. E. P. Davies, N. Degara, and M. D. Plumbley. Evaluation methods for musical audio beat tracking algorithms. *Queen Mary University of London, Tech. Rep. C4DM-09-06*, 2009.

T. De Clercq and D. Temperley. A corpus analysis of rock harmony. *Popular Music*, 30 (01):47–70, 2011.

N. Degara, E. Argones Rua, A. Pena, S. Torres-Guijarro, M. E. P. Davies, and M. D. Plumbley. Reliability-informed beat tracking of musical signals. *IEEE Transactions on Audio, Speech, and Language Processing*, (99):1–1, 2011.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, E. Battenberg, A. van den Oord, et al. Lasagne: First release, 2015.

S. Dixon, F. Gouyon, and G. Widmer. Towards characterisation of music via rhythmic patterns. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, Barcelona, 2004.

S. Dixon, W. Goebl, and E. Cambouropoulos. Perceptual smoothness of tempo in expressively performed music. *Music Perception*, 23(3):195–214, 2006.

R. Douc, O. Cappé, and E. Moulines. Comparison of resampling schemes for particle filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 64–69. IEEE, 2005.

A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 2009.

C. Drake and M. Botte. Tempo sensitivity in auditory sequences: Evidence for a multiple-look model. *Perception & Psychophysics*, 54(3):277–286, 1993.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

S. Durand, B. David, and G. Richard. Enhancing downbeat detection when facing different music styles. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3132–3136. IEEE, 2014.

S. Durand, J. Bello, D. Bertrand, and G. Richard. Downbeat tracking with multiple features and deep neural networks. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, 2015.

S. Durand, J. P. Bello, Bertrand D., and G. Richard. Feature adapted convolutional neural networks for downbeat tracking. In *The 41st IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

G. Dzhambazov. Towards a drum transcription system aware of bar position. In *Proceedings of the AES 53rd International Conference on Semantic Audio*, London, 2014.

D. Eck. Beat tracking using an autocorrelation phase matrix. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 4, pages 1313–1316, Honolulu, April 2007.

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

F. Eyben, B. Schuller, S. Reiter, and G. Rigoll. Wearable assistance for the ballroom-dance hobbyist-holistic rhythm analysis and dance-style classification. In *Proceedings of the 8th IEEE International Conference on Multimedia and Expo (ICME)*, Beijing, 2007.

T. Fillon, C. Joder, S. Durand, and S. Essid. A conditional random field system for beat tracking. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 424–428. IEEE, 2015.

B. D. Giorgi, M. Zanoni, A. Sarti, and S. Tubaro. Automatic chord recognition based on the probabilistic modeling of diatonic modal harmony. In *Proceedings of the 8th International Workshop on Multidimensional Systems*, 2013.

S. Godsill, A. Doucet, and M. West. Maximum a posteriori sequence estimation using Monte Carlo particle filters. *Annals of the Institute of Statistical Mathematics*, 53(1): 82–96, 2001.

M. Goto. An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30(2):159–171, 2001.

M. Goto. AIST annotation for the RWC music database. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, pages 359–360, Victoria, 2006.

M. Goto and Y. Muraoka. Real-time rhythm tracking for drumless audio signals: Chord change detection for musical decisions. *Speech Communication*, 27(3):311–335, 1999.

M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Popular, classical and jazz music databases. In *Proceedings of the 3rd International Society for Music Information Retrieval Conference (ISMIR)*, Paris, 2002.

F. Gouyon. *A computational approach to rhythm description-Audio features for the computation of rhythm periodicity functions and their use in tempo induction and music content processing*. PhD thesis, Universitat Pompeu Fabra, 2005.

F. Gouyon. The ballroom dataset. `http://mtg.upf.edu/ismir2004/contest/tempoContest/node5.html`, 2006. Online; Accessed: 2016-09-18.

F. Gouyon, S. Dixon, E. Pampalk, and G. Widmer. Evaluating rhythmic descriptors for musical genre classification. In *Proc. of the AES 25th International Conference*, pages 196–204, London, 2004.

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

A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868, 2009.

S. Hainsworth and M. Macleod. Particle filtering applied to musical tempo tracking. *EURASIP Journal on Applied Signal Processing*, 2004:2385–2395, 2004.

A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Ng. Deep speech: Scaling up end-to-end speech recognition. *arXiv:1412.5567v2*, 2014.

Y. Hochberg and A. Tamhane. *Multiple comparison procedures*. John Wiley & Sons, Inc., 1987.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

J. Hockman, M. E. P. Davies, and I. Fujinaga. One in the jungle: Downbeat detection in hardcore, jungle, and drum and bass. In *Proceedings of the 13th International Society for Music Information Retrieval (ISMIR)*, Porto, 2012.

A. Holzapfel and E. Benetos. The sousta corpus: Beat-informed automatic transcription of traditional dance tunes. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York, 2016.

A. Holzapfel and T. Grill. Bayesian meter tracking on learned signal representations. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York, 2016.

A. Holzapfel, M. E. P. Davies, J. Zapata, J. Oliveira, and F. Gouyon. Selective sampling for beat tracking evaluation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(9):2539–2548, 2012.

A. Holzapfel, F. Krebs, and A. Srinivasamurthy. Tracking the "odd": Meter inference in a culturally diverse music corpus. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, 2014.

T. Jehan. Downbeat prediction by listening and learning. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 267–270. IEEE, 2005.

A. Johansen and A. Doucet. A note on auxiliary particle filters. *Statistics & Probability Letters*, 78(12):1498–1504, 2008.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

M. Khadkevich, T. Fillon, Gaël R., and M. Omologo. A probabilistic approach to simultaneous extraction of beats and downbeats. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 445–448. IEEE, 2012.

G. Kitagawa. Monte carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1–25, 1996.

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

M. Kolinski. A cross-cultural approach to metro-rhythmic patterns. *Ethnomusicology*, 17(3):494–506, 1973.

F. Korzeniowski, S. Böck, and G. Widmer. Probabilistic extraction of beat positions from a beat activation function. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, 2014.

F. Krebs and S. Böck. MIREX 2012 audio beat tracking evaluation: NeuroBeatE. *Music Information Retrieval Evaluation eXchange (MIREX)*, 2012.

F. Krebs, S. Böck, and G. Widmer. Rhythmic pattern modeling for beat and downbeat tracking in musical audio. In *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, Curitiba, 2013.

F. Krebs, F. Korzeniowski, M. Grachten, and G. Widmer. Unsupervised learning and refinement of rhythmic patterns for beat and downbeat tracking. In *Proceedings of the 22nd European Signal Processing Conference (EUSIPCO)*, pages 611–615, 2014.

F. Krebs, S. Böck, and G. Widmer. An efficient state space model for joint tempo and meter tracking. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Malaga, 2015a.

F. Krebs, A. Holzapfel, A. T. Cemgil, and G. Widmer. Inferring metrical structure in music using particle filters. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(5):817–827, 2015b.

F. Krebs, S. Böck, M. Dorfer, and G. Widmer. Downbeat tracking using beat-synchronous features and recurrent neural networks. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, New York, 2016.

D. Lang and N. de Freitas. Beat tracking the graphical model way. *Advances in Neural Information Processing Systems (NIPS)*, 17:745–752, 2004.

J. Laroche. Efficient tempo and beat tracking in audio recordings. *Journal of the Audio Engineering Society*, 51(4):226–233, 2003.

F. Lerdahl and R. Jackendoff. *A generative theory of tonal music*. Cambridge: MIT Press., 1987.

P. Manuel. The anticipated bass in cuban popular music. *Latin American music review*, 6(2):249–261, 1985.

M. Mauch and S. Dixon. Simultaneous estimation of chords and musical context from audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1280–1289, 2010.

T. P. Minka. From hidden Markov models to linear dynamical systems. *Department of Electrical Engineering and Computer Science, MIT, Cambridge, Mass, USA*, 1999.

M. Müller and S. Ewert. Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, Miami, 2011.

M. Müller, D. P. W. Ellis, A. Klapuri, and G. Richard. Signal processing for music analysis. *IEEE Journal of Selected Topics in Signal Processing*, 5(6):1088–1110, 2011.

K. Murphy. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002.

J. L. Oliveira, M. E. P. Davies, F. Gouyon, and L. P. Reis. Beat tracking for multiple applications: A multi-agent system architecture with state recovery. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(10):2696–2706, 2012.

H. Papadopoulos and G. Peeters. Joint estimation of chords and downbeats from an audio signal. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(1): 138–152, 2011.

R. Parncutt. A perceptual model of pulse salience and metrical accent in musical rhythms. *Music Perception: An Interdisciplinary Journal*, 11(4):409–464, 1994.

G. Peeters and H. Papadopoulos. Simultaneous beat and downbeat-tracking using a probabilistic framework: theory and large-scale evaluation. *IEEE Transactions on Audio, Speech, and Language Processing*, (99):1–1, 2011.

M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American statistical association*, 94(446):590–599, 1999.

T. Pohle, D. Schnitzer, M. Schedl, P. Knees, and G. Widmer. On rhythm and general music similarity. In *Proceedings of the 10th International Society for Music Information Retrieval (ISMIR)*, Kobe, 2009.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

L. R. Rabiner, J. G. Wilpon, and B. Juang. A segmental k-means training procedure for connected word recognition. *AT&T Technical Journal*, 65(3):21–31, 1986. ISSN 1538-7305. doi: 10.1002/j.1538-7305.1986.tb00368.x.

E. Ramasso. Contribution of belief functions to hidden markov models with an application to fault diagnosis. In *IEEE International Workshop on Machine Learning for Signal Processing, 2009. MLSP 2009*, pages 1–6, September 2009.

L. J. Rodríguez and I. Torres. Comparative study of the baum-welch and viterbi training algorithms applied to read and spontaneous speech recognition. In *Pattern Recognition and Image Analysis*, page 847–857. Springer, 2003.

S. Roweis and Z. Ghahramani. A unifying review of linear Gaussian models. *Neural computation*, 11(2):305–345, 1999.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

X. Serra, M. Magas, E. Benetos, M. Chudy, S. Dixon, A. Flexer, E. Gómez, F. Gouyon, P. Herrera, S. Jorda, O. Paytuvi, G. Peeters, J. Schlüter, H. Vinet, and G. Widmer. *Roadmap for Music Information ReSearch*. Creative Commons BY-NC-ND 3.0 license, 2013. ISBN 978-2-9540351-1-6.

W. A. Sethares, R. D. Morris, and J. C. Sethares. Beat tracking of musical performances using low-level audio features. *IEEE Transactions on speech and audio processing*, 13 (2):275–285, 2005.

U. Şimşekli, O. Sönmez, B. Kurt, and A. T. Cemgil. Combined perception and control for timing in robotic music performances. *EURASIP Journal on Audio, Speech, and Music Processing*, 2012(1):1–20, 2012.

A. Srinivasamurthy and X. Serra. A supervised approach to hierarchical metrical cycle tracking from audio music recordings. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5217–5221, Florence, 2014.

A. Srinivasamurthy, A. Holzapfel, and X. Serra. In search of automatic rhythm analysis methods for Turkish and Indian art music. *Journal for New Music Research*, 43(1): 94–114, 2014.

A. Srinivasamurthy, A. Holzapfel, A. T. Cemgil, and X. Serra. Particle filters for efficient meter tracking with dynamic Bayesian networks. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Malaga, 2015.

A. Srinivasamurthy, A. Holzapfel, A. T. Cemgil, and X. Serra. A generalized bayesian model for tracking long metrical cycles in acoustic music signals. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 76–80. IEEE, 2016.

H. Stobart and I. Cross. The Andean anacrusis: Rhythmic structure and perception in easter songs of northern Potosí, Bolivia. *British Journal of Ethnomusicology*, 9(2): 63–92, 2000.

B. Sturm. Faults in the ballroom dataset. http://media.aau.dk/null_space_pursuits/2014/01/ballroom-dataset.html, 2014. Online; Accessed: 2016-04-14.

J. Vermaak, A. Doucet, and P. Pérez. Maintaining multimodality through mixture tracking. In *Proceedings of the 9th IEEE International Conference on Computer Vision*, pages 1110–1116. IEEE, 2003.

A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.

N. Whiteley, A. Cemgil, and S. Godsill. Bayesian modelling of temporal structure in musical audio. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, Victoria, 2006.

N. Whiteley, A. T. Cemgil, and S. Godsill. Sequential inference of rhythmic structure in musical audio. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages IV–1321, Honolulu, 2007.

I. Winkler, GP Háden, O. Ladinig, I. Sziller, and H. Honing. Newborn infants detect the beat in music. *Proceedings of the National Academy of Sciences*, 106(7):2468–2471, 2009.

J. Zapata, M. Davies, and E. Gómez. Multi-feature beat tracking. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 22(4):816–825, 2014.

Curriculum Vitae of the Author

Dipl.-Ing. Florian Krebs

Year of Birth:	1980
Nationality:	German
Email:	Florian.Krebs@jku.at
Homepage:	http://www.cp.jku.at/people/krebs

Main areas of research

- Music information retrieval
- Audio signal processing
- Machine learning

Academic career

2011-2016	Researcher at the Department of Computational Perception, Johannes Kepler University Linz
2013	Research visit at the Department of Computer Engineering, Boğaziçi University Istanbul

2002-2010	Diploma studies in electrical engineering and sound engineering (passed with distinction), Graz University of Technology and University of Music and Performing Arts Graz
2006	Studies in electrical engineering and music theory, Universidad Cathólica and Universidad Nacional de Córdoba (Argentina)
2000-2008	Studies in medicine, University of Regensburg and Medical University Graz

Teaching

2014-2016	Lecturer of the course *Probabilistic Graphical Models*, Johannes Kepler University Linz
2009-2010	Tutor of the course *Signal Processing*, Graz University of Technology

Awards, grants and prizes

2012-2016	**Audio beat tracking** evaluation exchange (MIREX): 3 times first place, two times second place and four times third place.
2012	**Audio tempo extraction** evaluation exchange (MIREX): First place (both tempi correct)
2014-2016	**Audio downbeat estimation** evaluation exchange (MIREX): 9 times first place.
2014	**Best paper award** at the AES 53rd international conference for the paper *Bridging the audio-symbolic gap: The discovery of repeated note content directly from polyphonic music audio* by T. Collins, S. Böck, F. Krebs, and G. Widmer
2014	**Travel grant** for the *International Workshop on Cross-disciplinary and Multi-cultural Perspectives on Musical Rhythm and Improvisation* at New York University Abu Dhabi.

Scientific services

Reviewer for international journals such as *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *EURASIP Journal on Advances in Signal Processing*, and *IEEE Signal Processing Letters*.

Reviewer for international conferences such as the *International Society for Music Information Retrieval Conference (ISMIR)*, and the *European Signal Processing Conference (EUSIPCO)*.

Task captain in the MIREX evaluation exchange from 2014 until 2016.

Publications

Journal publications

F. Krebs, A. Holzapfel, A. T. Cemgil, and G. Widmer. Inferring metrical structure in music using particle filters. In *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(5):817–827, 2015.

A. Lambert and F. Krebs. The Second International Workshop on Cross-Disciplinary and Multicultural Perspectives on Musical Rhythm and Improvisation. In *Computer Music Journal*, 2015.

M. Grachten, and F. Krebs. An Assessment of Learned Score Features for Modeling Expressive Dynamics in Music. In *IEEE Transactions on Multimedia: Special Issue on Data Mining*, 2014.

Peer-reviewed conference publications

F. Krebs, S. Böck, M. Dorfer, and G. Widmer. Downbeat tracking using beat- synchronous features and recurrent neural networks. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, New York, 2016.

S. Böck, F. Krebs, and G. Widmer. Joint Beat and Downbeat Tracking with Recurrent Neural Networks. In *Proceedings of 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York, USA, 2016.

S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, G. Widmer. madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia (MM)*, Amsterdam, The Netherlands, 2016.

F. Krebs, S. Böck, and G. Widmer. An efficient state space model for joint tempo and meter tracking. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Malaga, 2015.

S. Böck, F. Krebs, and G. Widmer. Accurate Tempo Estimation Based on Recurrent Neural Networks and Resonating Comb Filters. In *Proceedings of 16th International Society for Music Information Retrieval Conference (ISMIR)*, Malaga, Spain, 2015.

S. Böck, F. Krebs, and G. Widmer. A Multi-Model Approach to Beat Tracking Considering Heterogeneous Music Styles. In *Proceedings of 15th International Society for Music Information Retrieval Conference (ISMIR)*, Taipeh, Taiwan, 2014.

A. Holzapfel, F. Krebs, and A. Srinivasamurthy. Tracking the 'Odd': Meter Inference in a Culturally Diverse Music Corpus. In *Proceedings of 15th International Society for Music Information Retrieval Conference (ISMIR)*, Taipeh, Taiwan, 2014.

F. Krebs, F. Korzeniowski, M. Grachten, and G. Widmer. Unsupervised learning and refinement of rhythmic patterns for beat and downbeat tracking. In *Proceedings of the 22nd European Signal Processing Conference (EUSIPCO)*, 2014.

F. Korzeniowski, F. Krebs, A. Arzt, and G. Widmer. Tracking Rests and Tempo Changes: Improved Score Following with Particle Filters. In *Proceedings of the International Computer Music Conference (ICMC)*, Perth, Australia, 2013.

F. Krebs, S. Böck, and G. Widmer. Rhythmic pattern modeling for beat and downbeat tracking in musical audio. In *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, Curitiba, 2013.

F. Krebs and M. Grachten. Combining Score and Filter Based Models to Predict Tempo Fluctuations in Expressive Music Performances. In *Proceedings of the 9th Sound and Music Computing Conference (SMC)*, Copenhagen, Denmark, 2012.

S. Böck, A. Arzt, F. Krebs, and M. Schedl. Online Real-time Onset Detection with Recurrent Neural Networks. In *Proceedings of the 37th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Kyoto, Japan, 2012.

S. Böck, F. Krebs, and M. Schedl. Evaluating the online capabilities of onset detection methods. In *Proceedings of the 14th International Conference on Music Information Retrieval (ISMIR)*, Porto, 2012.