The Language of Music: A Computational Model of Music Interpretation

Andrew McLeod

Doctor of Philosophy
Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh
2018

Abstract

Automatic music transcription (AMT) is commonly defined as the process of converting an acoustic musical signal into some form of musical notation, and can be split into two separate phases: (1) multi-pitch detection, the conversion of an audio signal into a time-frequency representation similar to a MIDI file; and (2) converting from this time-frequency representation into a musical score. A substantial amount of AMT research in recent years has concentrated on multi-pitch detection, and yet, in the case of the transcription of polyphonic music, there has been little progress.

There are many potential reasons for this slow progress, but this thesis concentrates on the (lack of) use of music language models during the transcription process. In particular, a music language model would impart to a transcription system the background knowledge of music theory upon which a human transcriber relies. In the related field of automatic speech recognition, it has been shown that the use of a language model drawn from the field of natural language processing (NLP) is an essential component of a system for transcribing spoken word into text, and there is no reason to believe that music should be any different.

This thesis will show that a music language model inspired by NLP techniques can be used successfully for transcription. In fact, this thesis will create the blueprint for such a music language model. We begin with a brief overview of existing multi-pitch detection systems, in particular noting four key properties which any music language model should have to be useful for integration into a joint system for AMT: it should (1) be probabilistic, (2) not use any data a priori, (3) be able to run on live performance data, and (4) be incremental.

We then investigate voice separation, creating a model which achieves state-of-the-art performance on the task, and show that, used as a simple music language model, it improves multi-pitch detection performance significantly. This is followed by an investigation of metrical detection and alignment, where we introduce a grammar crafted for the task which, combined with a beat-tracking model, achieves state-of-the-art results on metrical alignment. This system's success adds more evidence to the long-existing hypothesis that music and language consist of extremely similar structures.

We end by investigating the joint analysis of music, in particular showing that a combination of our two models running jointly outperforms each running independently. We also introduce a new joint, automatic, quantitative metric for the complete transcription of an audio recording into an annotated musical score, something which the field currently lacks.

Lay Summary

Automatic music transcription (AMT) is commonly defined as the process of converting an musical recording into some form of musical notation, and can be split into two phases: (1) the detection of the musical notes present in the recording; and (2) converting from these notes into a musical score. A substantial amount of research in AMT in recent years has concentrated on the first phase, and yet, in the case of musical recordings which contain multiple simultaneous notes, there has been little progress.

There are many potential reasons for this slow progress, but this thesis concentrates on the (lack of) use of musical knowledge during the transcription process. A human performing the task would rely on a background knowledge of music theory, but most AMT systems do not have a way to use such knowledge. In the related area of the transcription of the spoken word into text, it has been shown that the use of knowledge about the structure and meaning of natural language (in the form of a language model) is an essential component of a successful system, and there is no reason to believe that music should be any different.

This thesis will show that musical knowledge can be imparted to music transcription systems with ideas inspired by natural language research through the use of a music language model. In fact, this thesis will create the blueprint for how such a music language model should be structured. We begin with a brief overview of existing music transcription systems, noting four key properties which any music language model should have should have to be useful for integration into a complete AMT system.

We then investigate the creation of music language models covering multiple aspects of music. First, we look at models which can separate a list of notes into their individual parts (for example, by which instrument made each note), and we learn that the use of such a model improves the performance of transcription of an audio recording significantly. Next, we investigate the modelling of rhythmic aspects of music, and learn that musical rhythms have a very similar structure to natural language by showing that a type of language model known as a grammar can detect the underlying structure of a musical performance.

Finally, we end by combining our two models, showing that their combination, run together, outperforms each running separately. We also introduce a new metric to judge the performance of any model performing a complete transcription of a musical recording into a musical score, something that is severely lacking in the field.

Acknowledgements

There are countless people, far too many to name them all, without whom this thesis wouldn't be what it is today. First and foremost, my supervisor, Mark Steedman. It has been a great pleasure to work with Mark over the years, and I have learned a great deal from our frequent conversations about music, both related and unrelated to this thesis. The perspective I have gained from Mark during my time at Edinburgh has been invaluable.

I would like to thank my viva examiners David Temperley and Chris Lucas for their feedback regarding my initial submitted version of this thesis. I am sure that the changes derived from our discussion have made this a more complete work.

I would like to thank Rodrigo Schramm and Emmanouil Benetos for their collaboration on the acoustic voice assignment model in Chapter 3. In particular, Rodrigo and I spent a great deal of time together working on the code to integrate our models, and I owe both of them thanks for inviting me on my numerous visits to Queen Mary University of London to aid in the collaboration.

Emmanouil deserves additional thanks for his feedback regarding an early version of the new evaluation metric in Chapter 5, as well as his time and expertise in discussing my future plans, including his work on our joint grant proposal.

I would like to thank Nathan Schneider, whose advice and feedback at the beginning of my PhD, both about my research, and about music and academia in general have been invaluable to this process. Our weekly lunch conversations at Nawroz led directly to me organising a workshop at Edinburgh at which my collaboration with Rodrigo and Emmanouil first began to take shape.

I would like to thank James Owers for our weekly golf "meetings" which have kept me sane over these past few months, as well as his discussions with me on music and research in general, and his feedback on earlier, much worse drafts of this thesis.

I would like to thank my friends, both in Edinburgh and around the world. A PhD is a long process, and their support and conversations have seen me through this chapter of my life with a smile on my face.

I would like to thank Steve Nowicki, who was truly the inspiration for me to start on this path towards academia. I still vividly remember our conversation at Chocolat where I finally took seriously his suggestion to consider it as a real option.

Last, but certainly not least, I would like to thank my parents and my sister for their unconditional love and support over these past 27 years. I certainly could not have done any of this without each one of you.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Andrew McLeod*)

"The written word seems deceptively free from the ebb and flow of feeling and emotion, but the spoken word defies separation from the flow of consciousness of the speaker. Perhaps, then, the problem has been wrongly posed. We should be asking not, how is it that music can have meaning when there are no words to convey it, but, how can the written word convey meaning when there is no indication how it should be uttered? The wonder should be, not that we can communicate through song, dance and gesture, but that we can exchange experiences so effectively through such a dry-as-dust-medium as patterns of marks on processed wood pulp."

—*The Language of Music* (Longuet-Higgins, 1972)

Table of Contents

Chapter 1

Introduction

The most common definition of automatic music transcription (AMT) is quite vague, referring to the ability to take as input an audio signal and output "some form of music notation" (Benetos, Dixon, Giannoulis, Kirchhoff, & Klapuri, 2013). We define the "complete" AMT problem more specifically to refer to the conversion of an audio signal into a complete musical score.

Conceptually, it can be useful to split the complete transcription process into two separate phases: (1) multiple fundamental frequency detection (multi-pitch detection), the conversion of an audio signal into a time-frequency representation, often going as far as detecting note-events and assigning each note a pitch, onset time, and offset time, as in a MIDI file; and (2) converting from this time-frequency representation into a musical score by performing musical analysis such as voice separation, metrical analysis, and harmonic analysis. These two steps of the complete AMT pipeline are illustrated in Figure 1.1.

A wealth of research has been performed on the first phase, multi-pitch detection. In fact, for monophonic signals—those containing only one note at a time—the problem is considered solved. However, the accuracy of even state-of-the-art multi-pitch detection models on polyphonic signals is still clearly below that of a human expert performing the same task (Benetos et al., 2013). Also concerning is the fact that there have been only minimal frame-based accuracy improvements in the corresponding annual shared task: the Music Information Retrieval Evaluation Exchange (MIREX) Multiple Fundamental Frequency Estimation & Tracking Task (MIREX, 2017e), as shown in Figure 1.2. In fact, between 2010 and 2017 (the most recent year for which results are currently available), the highest accuracy was 0.72 by Elowsson and Friberg (2014) and by Thickstun, Harchaoui, and Kakade (2017), narrowly beating out the 0.69 accu-

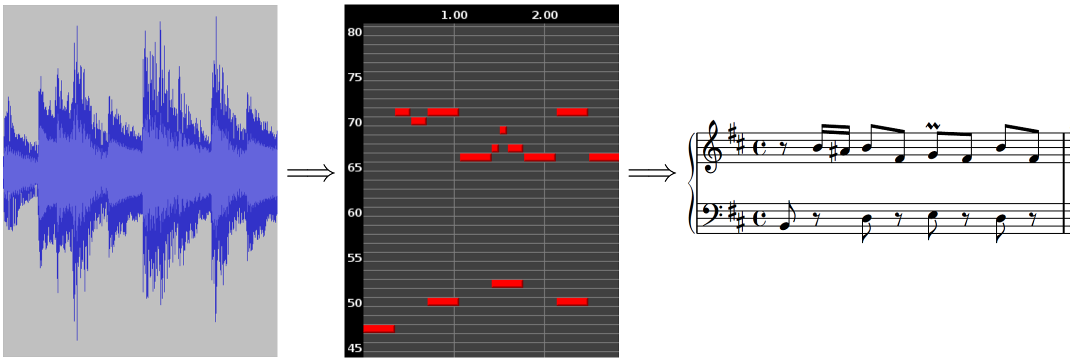


Figure 1.1: The two steps of the complete AMT pipeline. First, multi-pitch detection is performed to convert an input audio recording (left) into some time-frequency representation (here, MIDI data, middle). Next, further musical analysis is performed to convert the MIDI data into a musical score (right).

racy by Yeh (2008) in 2010. Note that the dataset has remained unchanged throughout this period

One possible reason for this lack of progress is that the multi-pitch detection models have not tended to use the background knowledge of music theory upon which a human transcriber relies. In the field of automatic speech recognition, it has been shown that a background language model drawn from natural language processing (NLP) is essential to the transcription of audio into text (Young, 1996), and there is no reason to think that music should be any different. In fact, music and language are much more similar than might seem at first glance.

Meyer (1956), for example, describes emotion and meaning in music, comparing it to emotion and meaning of other forms of communication. Bernstein (1976) takes this comparison further, hypothesising numerous parallels and analogies between music and different aspects of language, from phonemes, to words, to sentences and beyond. Lerdahl and Jackendoff (1985) formalise this comparison, proposing a grammar for the hierarchical analysis of music based on grammars created for the analysis of natural language used in linguistics and NLP. More recently, a variety of work has used NLP-based grammars to parse harmonic progressions in a variety of Western music, demonstrating that the harmonic progressions of Western tonal music have a syntactic structure similar to that of natural language (Steedman, 1996; Rohrmeier, 2011; Granroth-Wilding & Steedman, 2014).

Some AMT systems have begun to incorporate limited musicological information in the form of a simple music language model which runs either jointly with the acous-

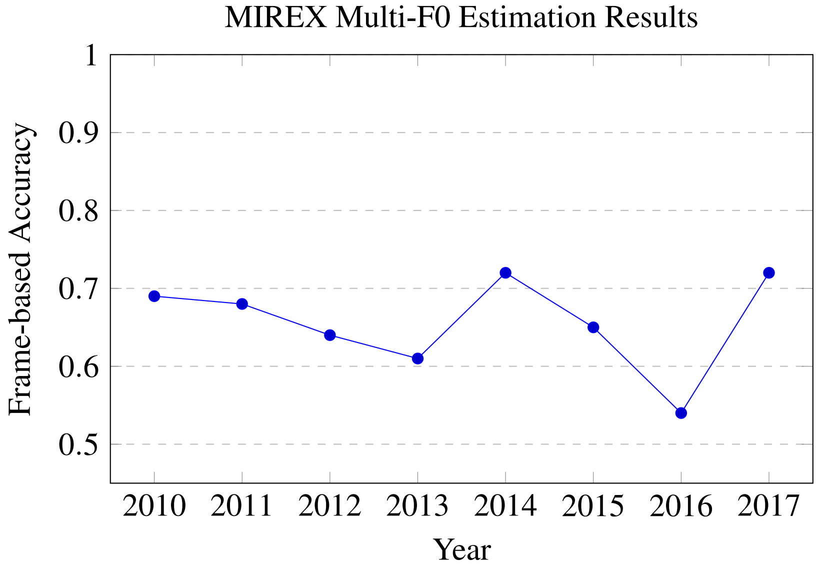


Figure 1.2: The frame-based accuracy of the top-scoring model in the MIREX Multi-F0 Estimation task for each year since 2010 on the MIREX dataset (which has remained unchanged).

tic multi-pitch detection model, or as a post-processing step. Berg-Kirkpatrick, Andreas, and Klein (2014), for example, use a form of simple music language modelling where each note is assigned its own music language model to guide its pitch activations. Sigtia, Benetos, and Dixon (2016) use a recurrent neural network (RNN) music language model along with a convolutional neural network (CNN) acoustic model for the transcription of piano recordings. These models have indeed shown some improvement over previous AMT systems (although their results cannot be compared to the MIREX results since they are both evaluated specifically on piano music). Even so, it is clear that the music language models seem to be limited to learning the low-level structure of music at the note level or lower—analogous to phoneme-level modelling in speech recognition—rather than any more insightful music analysis. A complete music language model has yet to be offered.

This thesis demonstrates that a music language model, inspired by NLP techniques, can be used successfully for higher-level music language modelling. In doing so, it adds further evidence in support of the hypothesised similarities between music and natural language. It also proves that the addition of such a language model can significantly improve the accuracy of acoustic models for multi-pitch detection, and that modelling different aspects of music jointly adds value over independent modelling. To that end, it outlines desirable properties that a music language model should have for it to be useful in a joint model for improving AMT performance, and it proposes a new evaluation metric for the task of complete AMT.

It is important to note that although the models presented in this thesis are often inspired by human cognition (since humans outperform computers on many transcription tasks), they should not be seen as models of human cognition. They do indeed both run on the same input information as a human would receive (a non-annotated stream of notes or audio) and perform their analyses incrementally as a human would. However, we do not argue that humans necessarily interpret or analyse music in a similar fashion to our models, nor do we evaluate our models with human cognition in mind. Rather, the focus of this thesis is on the improvement AMT performance, and the models are evaluated in such a context.

1.1 Data Formats

This thesis will use data in a variety of formats, including audio recordings as well as more data formats. This section gives an overview of each.

Audio recordings are the most natural format for storing music performance, consisting of a digital representation of a waveform. There are many different audio file types; however, regardless of the specific file type or compression strategy used, each essentially contains the same information: the value of the recorded waveform at some time interval.

The main input to the models presented in this thesis, however, is not audio data. It is instead MIDI data, which contains a more abstract representation of music. We use the term MIDI data to refer to any data format which contains a list of musical notes, where each is assigned a pitch (in equally-tempered semitone scale), an onset time, and an offset time, such as a piano roll. In practice, many different formats are equivalent to MIDI in this regard. Additional information such as voicing, metrical structure, and key signature are also sometimes included in MIDI data, though we ignore them for the purposes of our analyses. This gives the added benefit of allowing the models presented in this thesis to be run directly on the wealth of non-annotated MIDI data which is available online.

There are two types of MIDI data: (1) metronomic and (2) live performance. In metronomic data, the notes are a perfectly metronomic representation of the notes of a musical score, though the meter, tempo, and key signature are not always annotated. Such data is a representation of a musical piece rather than any specific performance of that piece, and is usually computer generated. Live performance data, on the other hand, does indeed represent a specific performance of a piece, and is either generated by hand from audio data, or recorded directly from an instrument capable of producing it, such as a MIDI keyboard.

Note that although MIDI is a widely used format, it has its limitations. While it has the ability to represent performance techniques like vibrato or portamenti (slides) as control change (CC) messages, these are only used in practice when the MIDI file is recorded directly from a MIDI instrument. For other MIDI files, such CC messages are rarely present, although they are often annotated in musical scores. Thus, for a complete AMT system, further processing must be performed in addition to any analyses which can be learned from MIDI files directly (such as those presented in this thesis).

1.2 Thesis Overview

Chapter 2 presents a brief overview of existing multi-pitch detection models which take an audio recording as input and produce some MIDI-like format as output. While multi-pitch detection itself is not the explicit subject of any section of this thesis (and thus, such an overview would not fit directly into one of the other chapters), a strong knowledge of the current state of the field as a whole gives important context to the remaining chapters. In particular, Chapter 2 will enumerate four key properties which models created for the purpose of improving AMT (such as those created in this thesis) should have.

Chapter 3 describes work on voice separation and assignment, the process of taking a polyphonic list of notes and separating them into monophonic streams. Voice separation is often performed simply as a preprocessing step for further music analysis; however, we will show that a good model for voice separation, used as a simple music language model, can be integrated with an acoustic model to significantly improve multi-pitch detection as well as voice assignment from acoustic input. Thus, we will show that voice separation should in future play an important role in music language modelling.

Chapter 4 contains work on the metrical analysis of music, in particular the detection and alignment of the underlying metrical structure of an input stream of notes. We will show that the use of a lexicalised probabilistic context-free grammar (LPCFG), a type of tree grammar also used in NLP, is able to capture the complex dependencies of musical rhythms and detect this underlying metrical structure. That such grammars have also been used for NLP tasks suggests that the structures of music and language are in some way similar. Thus, we will show that the use of NLP techniques for music language modelling is a promising approach worthy of future research.

Chapter 5 discusses the joint modelling of multiple aspects of music, where we will show that such an approach offers improved results compared with independently modelling each component. Thus, we will demonstrate that even seemingly unrelated components of musical analysis have the potential to inform and improve each other, suggesting that music language models should be designed to make joint analysis possible. Towards the goal of joint music language modelling for AMT, we will also suggest a new metric for the complete transcription and analysis of musical audio—something which is lacking in a field that will likely be the subject of much research in the years to come.

Finally, general conclusions and ideas for future work are presented in Chapter 6.

Chapter 2

Multi-pitch Detection

This chapter contains an overview of existing work on multi-pitch detection. The main goal of this thesis, to create a music language model for the purpose of improving acoustic transcription, does not directly involve multi-pitch detection per se. However, a future plan is to integrate our full music language model with a multi-pitch detection system such as any of those described here. Thus, while multi-pitch detection itself is not the explicit subject of this thesis, a strong knowledge of state-of-the-art approaches in the field helps to give context to the remaining chapters, in particular to highlight what properties and constraints our music language model (as well as the multi-pitch detection model) must have in order for such integration to be possible.

Furthermore, in Chapter 3, we present a joint system for multi-pitch detection and voice assignment, combining an acoustic multi-pitch detection model based on probabilistic latent component analysis (PLCA; see Section 2.2) with a voice separation music language model, and a knowledge of existing multi-pitch detection models will also be valuable as background to that work.

2.1 Introduction

A large majority of methods for multi-pitch detection in recent years have been based on some form of spectrogram analysis. A spectrogram is a time-frequency representation of an audio signal where a single waveform is converted into a matrix whose value in each cell represents the strength of a single frequency (corresponding to the row) at a specific time (corresponding to the column). The time (horizontal) axis is usually on a linear scale, with each column representing an equal-length duration called a frame, while the frequency (vertical) axis is usually either linear or logarithmic, with each

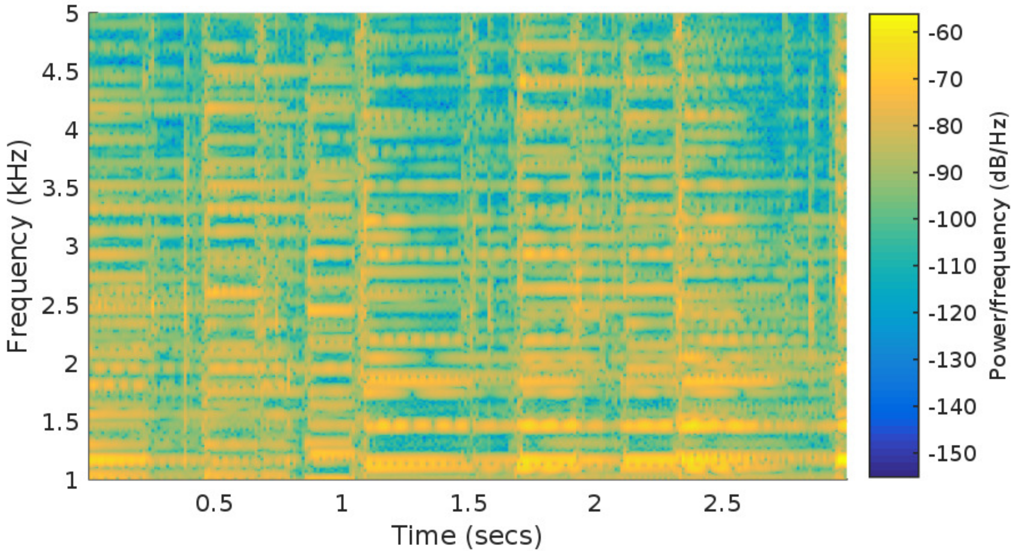


Figure 2.1: The spectrogram of a 3-second audio clip.

row representing a particular frequency range. The most basic method to calculate a spectrogram from a waveform is the Fourier Transform, resulting in a linear frequency scale. Convolutional neural networks (CNNs) were also used by Bittner, McFee, Salamon, Li, and Bello (2017) for learning a pitch salience representation—essentially a spectrogram where noise content is reduced and harmonic pitches are emphasised—for multi-pitch detections in polyphonic audio recordings. Figure 2.1 shows an example spectrogram.

This chapter presents an overview of methods for multi-pitch detection organised by their basic approach. Spectrogram factorisation methods are described in Section 2.2, deep learning approaches are described in Section 2.3, and other methods are discussed in Section 2.4. Note tracking, the process of converting frame-based multi-pitch detections into a sequence of musical notes, a necessary step for any complete AMT system, is discussed in Section 2.5. A conclusion, which puts the described methods into the larger context of this thesis, highlighting desirable features for our music language model as well as for a multi-pitch detection model, is presented in Section 2.6.

2.2 Spectrogram Factorisation

Spectrogram factorisation involves decomposing an input time-frequency representation (such as a spectrogram) into a linear combination of factors. Such methods have been used extensively for multi-pitch detection in the last decade, the most successful of which have been based on non-negative matrix factorisation (NMF; D. D. Lee & Seung, 1999) or probabilistic latent component analysis (PLCA; Shashanka, Raj, & Smaragdis, 2008).

NMF is based on the assumption that an object is well-represented as a combination of parts. For example, a single spectrogram consisting of many notes could be seen as a combination of the spectrograms of those notes individually. Thus, NMF attempts to decompose a spectrogram V (represented as a matrix) into the product of two matrices W and H, which are both constrained to be non-negative. This non-negativity constraint guides the decomposition towards the desired parts-based representation. Each column of W is a basis vector, and each row of H represents the contribution of the corresponding basis vector to the spectrogram across time. For a successful decomposition, each basis vector in W would ideally represent a single note; thus, the rows of H would correspond to the presence of that pitch across time in the input spectrogram. The factorisation is performed by minimising a cost function between V and WH, most often a simple Euclidean distance, shown in Equation (2.1), and minimisation is performed iteratively until convergence.

$$||V - WH||^2 = \sum_{ij} (V_{ij} - (WH)_{ij})^2 \tag{2.1}$$

PLCA is the probabilistic equivalent of NMF, where Kullback-Leibler (KL) divergence (also known as relative entropy), shown in Equation (2.2), is instead used as the cost function. For this to be valid, the spectrogram V is treated as a bivariate probability distribution $P(t, \omega)$ over time t and frequency ω, and thus must be normalised to sum to 1.[1] The factorisation is then reformulated as a product of conditionally independent probability distributions as shown in Equation (2.3) in its most basic form, where $P(t)$ is the probability of any pitch occurring at a given time (the spectral energy of the spectrogram at a given time), $P(\omega|p)$ is the dictionary containing the spectrum of every possible pitch (equivalent to W in NMF), and $P(p|t)$ is the probability of a pitch being present at a given time (equivalent to H in NMF). The expectation-maximisation (EM) algorithm (Dempster, Laird, & Rubin, 1977) is then used iteratively to minimise the KL divergence between this product and the true spectrogram until the model converges.

[1] Intuitively, the spectrogram can be interpreted as the distribution of the spectral energy of an audio signal across a range of frequencies and times.

$$D(V||WH) = \sum_{ij}(V_{ij}\log\frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij}) \tag{2.2}$$

$$P(\omega,t) = P(t)\sum_{p} P(\omega|p)P(p|t) \tag{2.3}$$

While these spectrogram factorisation methods have shown promise for multi-pitch detection in their most basic form, their parameter estimation can suffer from local optima, a problem which has motivated a variety of approaches that incorporate additional knowledge in an attempt to achieve more meaningful decompositions. Kameoka et al. (2012) exploit structural regularities in the spectrograms during the NMF process, adding constraints and regularisation to reduce the degrees of freedom of their model. These constraints are based on time-varying basis spectra (e.g., using sound states: "attack", "decay", "sustain", and "release"), and have since been included in other probabilistic models (Benetos & Dixon, 2013; Benetos & Weyde, 2015). Fuentes, Badeau, and Richard (2014) introducfe the concept of brakes, slowing the convergence rate of any model parameter known to be properly initialised.

For the factorisation, the basis vectors can either be pre-trained in a supervised manner, resulting in better performance (assuming the training data contains similar instruments and recording environments to the test data), or they can be learned adaptively from the input spectrogram, resulting in lower accuracy but better generalisation across different instruments and recording environments. Mysore and Smaragdis (2009), Grindlay and Ellis (2011), and Benetos and Weyde (2015), for example, all learn a dictionary of the spectral atoms of specific instruments during training. Vincent, Bertin, and Badeau (2010), on the other hand, use an adaptive spectral decomposition process, relying on the harmonicity of musical sounds to model pitches as combinations of narrowband spectra which preserve a smooth spectral envelope for the partials of a pitch.

Using the Constant-Q transform (CQT; J. Brown, 1991) to calculate the spectrogram results in a log-scaled frequency axis. To take advantage of this, some approaches develop techniques using shift-invariant models over log-frequency (Mysore & Smaragdis, 2009; Fuentes, Badeau, & Richard, 2012; Benetos & Dixon, 2012, 2013; Benetos, Badeau, Weyde, & Richard, 2014), allowing for the creation of a compact set of dictionary templates that can support tuning deviations and frequency modulations. O'Hanlon, Nagano, Keriven, and Plumbley (2016) propose stepwise and gradient-based methods for non-negative group sparse decompositions, exploring the use of subspace modelling of note spectra. This group sparse NMF approach is used to tune a generic harmonic subspace dictionary, improving automatic music transcription results based on NMF.

2.3 Deep Learning

Recent research on multi-pitch detection has also focused on deep learning approaches. For example, Böck and Schedl (2012) use a bidirectional long short-term memory (BLSTM) recurrent neural network (RNN) for the task of pitch and onset detection from audio, showing good performance and generalisation. Boulanger-Lewandowski, Bengio, and Vincent (2013) also use an RNN for multi-pitch detection, this time preferring a frame-based output without explicitly modelling onsets.

Sigtia et al. (2016) compare the use of feed-forward, recurrent, and convolutional neural networks (CNNs) for multi-pitch detection of piano music, in an end-to-end system consisting of both an acoustic model and a (frame-based) music language model, run on a CQT spectrogram. Their best performing model consists of a CNN acoustic model with a frame-based RNN music language model. The results with the RNN are only marginally better than those using simple thresholding or a hidden Markov model (HMM) music language model, suggesting that a more sophisticated, note-based music language model might be required to see larger improvements in performance. In fact, this frame-based RNN is essentially a sophisticated note tracking model (see Section 2.5 for an overview of note tracking methods).

Kelz et al. (2016) also compare neural network architectures (feed-forward and convolutional, as well as a combination of both) and input representations for multi-pitch detection. Their best performing model is run on a spectrogram with log-scale frequency as well as log-scale strength, and consists of a network with three convolutional layers followed by two feed-forward layers. They show that this input and network combination achieves state of the art results on the frame-based transcription of piano music.

The top performing model from the 2017 MIREX Multi-F0 Estimation task, by Thickstun, Harchaoui, Foster, and Kakade (2017), based on a model previously described by Thickstun, Harchaoui, and Kakade (2017), applies a CNN to a filterbank of 512 log-spaced frequencies for the task. The first layer of the network convolves along log-frequency, while the second layer is fully connected across time for each audio frame length of 1/3 of a second. Finally, a linear classifier is used for each pitch to determine if it is present.

2.4 Other Methods

Klapuri (2005) proposes a perceptually motivated model in which single pitches are detected and then removed from the original signal iteratively. Ryynanen and Klapuri (2005) extend that model with a music language model. They use two HMMs for each possible pitch: a note HMM, which contains an attack state, a sustain state, and a noise state; and a silence HMM, which contains only a single state representing a rest. The observed data for the HMMs are features calculated from a perceptually motivated acoustic model. The two HMMs of a single pitch may not both be active at the same time, and transitions between HMMs (including across pitches) are calculated by a music language model. The music language model first calculates the key of the piece, and then applies a simple bi-gram model to monophonic sequences of notes. The process is iterative: the most likely monophonic note sequence is transcribed during each iteration until the resulting monophonic sequence is only silence.

Klapuri (2006) calculates the salience of a candidate pitch detection as a weighted sum of the amplitudes of its harmonic partials. The method performs spectral whitening as preprocessing, a process which makes the method more robust to the varying timbral properties of sounds produced by different instruments.

Yeh (2008), the best scoring method from the MIREX Multi-F0 Estimation task from the years 2010–2013, is a generative model composed of three components: a noise model, which models the background noise of a signal as randomly fluctuating white noise; a source model, which detects potential harmonic pitches without explicitly modelling the amplitudes of the individual partials; and a source interaction model, which models the amplitude of each individual partial. A joint estimation approach is used to maximise the harmonicity and spectral smoothness of each candidate pitch detection, as well as the smoothness of the evolution of each partial amplitude over time.

Pertusa and Iñesta (2012) also use joint estimation, calculating the partial strengths of polyphonic combinations of pitches directly, measuring each combination's spectral smoothness and strength. Smoothing is also performed across adjacent frames to ensure temporal continuity.

Berg-Kirkpatrick et al. (2014) describe a three-component generative model for the transcription of piano music, somewhat similar to NMF. The components are an event model, an activation model, and a spectrogram model for each note on a piano. The event model, a Markov model, can be considered a simple music language model, and generates a sequence of notes and rests, each with a duration (measured in time steps) and a velocity (which is 0 for the rests). The activation model is similar to the H matrix from NMF, and generates the envelope of each played note based on the duration and velocity from the event model. The spectrogram model, analogous to the W matrix from NMF, generates the full spectrum of each note given its envelope. The spectra for each note are summed to generate the final spectrogram.

The best performing method from the 2014 MIREX Multi-F0 Estimation task, Elowsson and Friberg (2014), has unfortunately remained underspecified. However, the authors do state that they use a combination of multiple spectrograms at different resolutions. Their model transforms these spectrograms into a higher level representation with a machine learning framework, from which onsets and pitches are detected jointly.

2.5 Note Tracking

One additional step that is necessary after a frame-based multi-pitch detection is note tracking. Note tracking involves the grouping of frame-based pitch detections into notes spanning multiple frames, each with an onset, offset, and pitch. The most simple method for this is rule-based, as in Bello, Daudet, and Sandler (2006). Here a pitch that is present for multiple consecutive frames is initially classified as a single note. After this, consecutive notes which are separated by a rest of shorter than some threshold are joined together, and finally those remaining notes which are shorter than some other threshold are removed.

Poliner and Ellis (2007) use HMMs for note tracking, one per pitch, each with two states: on and off. These HMMs are run over the frame-based detections for each pitch, and work to produce realistic transcriptions, reducing note onsets and offsets (which correspond to HMM state transitions) to reasonable levels, according to some learned transition probabilities. More recently, feature-based approaches have been used for note tracking, classifying each potential note and rest as true or false. For example, support vector machines (SVMs) were used by Weninger, Kirst, Schuller, and Bungartz (2013), with NMF activation strengths used as features, and SVMs as well as other classification models were used by Valero-Mas, Benetos, and Iñesta (2016), using PLCA activation probabilities as well as a rule-based note tracking transcription as features.

It should also be noted that the RNN music language model used by Sigtia et al. (2016) (see Section 2.3) is essentially used as a sophisticated note tracking model, since it works on the frame level.

2.6 Conclusion

Regardless of the core method used for multi-pitch detection, it is clear from model performance that acoustic models which are integrated with music language models, even seemingly unsophisticated ones, seem to outperform those with no such musicological knowledge. This is in direct analogy to the problem of automatic speech recognition, in which a language model is an essential component of a successful model. All of the deep learning methods mentioned in Section 2.3 can be thought of as containing a simple music language model, in particular modelling the progression of pitch detections across frames, and it is likely that this feature accounts for some of their good performance. However, music language models which are designed to capture musicological structure at the note level (such as the one described in this thesis) are also desirable.

In theory, such a note-level music language model could be used with any probabilistic multi-pitch detection system directly as a post-processing step, and most of the above methods output a probability distribution over possible pitch detections. In practice, however, a tighter integration between the acoustic model and the music language model components which allows each to inform the other, should lead to increased performance.

Assuming we have some acoustic multi-pitch detection model which outputs a probability distribution over detected notes, a music language model should have the following properties in order for such a tight integration to be possible:

- It must be probabilistic, taking as input a probability distribution over the detected notes and producing as output a probability distribution over those same notes. That is, the music language model, given a potential note, should output the probability of that note being present. Such a probability value could be used either in discriminatively (by evaluating every detected note) or generatively (by evaluating every potential note at the next time-step, thereby guiding the acoustic model). In either case, the final probability of a note would be the product of the acoustic model's probability for that note and the music language model's probability for that note.

- It must not use any specific information about a piece of music besides what can be generated directly by the acoustic model. In particular, this rules out the use of any a priori information regarding the underlying music such as its alignment with a metrical structure, its key signature, or the number of voices present. A significant amount of prior work on music language modelling require some amount of this information a priori. Our proposed music language model, on the other hand, uses no such information, allowing it to be run on non-annotated data directly, an added benefit given the large amount of such data available online.
- Similarly, it must be able to run on live performance data. Live performance may contain timing deviations, or non-simultaneous notes which overlap slightly. A music language must be robust to such idiosyncrasies in order for it to be run on live performance data (or, in the future, jointly with an acoustic multi-pitch detection model).
- Ideally, it would be incremental, working causally from the beginning to the end of the piece. While this incrementality is not necessary, it would allow for simpler integration with multi-pitch detection models, enabling its output to be used as a prior for the acoustic model without necessitating that the acoustic model run to completion first. Incrementality also allows a model to be used for real-time tasks such as live accompaniment or improvisation.

Throughout this thesis, we propose components of a music language model to be used for transcription; importantly, our components all match the four criteria enumerated here. We will not apply our full model directly to audio files, but in Chapter 3, we introduce an integrated system using only a voice separation model as its music language model. That system's success strongly suggests that our eventual goal of creating a joint system consisting of an acoustic model with a fully-fledged music language model is both attainable and has the potential for success.

Chapter 3

Voice Separation

This chapter discusses voice separation, the division of the notes of a musical performance into streams called voices, and voice assignment, which additionally assigns a label to each voice representing the instrument or part to which those notes belong. Voice separation should be an integral component of any music language model, enabling the other components of the model to perform computation on multiple simple monophonic streams of notes, rather than a complicated polyphonic stream. This is analogous to a speech transcription system analysing multiple different recordings with one speaker each, rather than a single recording with multiple speakers speaking simultaneously. Additionally, voice separation and assignment are necessary components of any transcription system which is to be run on recordings containing multiple instruments, since each instrument is meant to be placed on an individual staff in the sheet music.

We design an HMM for the voice separation of live performance MIDI data which follows all of our constraints for a music language model from Chapter 2, being probabilistic, incremental, using no a priori information, and being able to run on live performance data. In fact, to our knowledge, our voice separation model is the first to have all four of these properties. Section 3.2 presents that HMM, and Section 3.3 combines an acoustic multi-pitch detection model with a modified version of it for the voice assignment of audio recordings of a cappella music, showing in practice that our model works alongside a multi-pitch detection system, and that voice separation is indeed a valuable component of a music language model.

This chapter is based on the published works "HMM-based voice separation of MIDI performance" (McLeod & Steedman, 2016), "Multi-pitch detection and voice assignment for a cappella recordings of multiple singers" (Schramm, McLeod, Steedman, & Benetos, 2017), and “Automatic transcription of polyphonic vocal music” (McLeod, Schramm, Steedman, & Benetos, 2017).

3.1 Introduction

Voice separation refers to the allocation of the notes of a given piece of music into streams of notes called voices, while voice assignment necessitates the additional step of labelling each voice as belonging to a specific part or instrument. It can be difficult to precisely define what constitutes a musical voice, and interested readers should refer to Cambouropoulos (2008) for a full discussion on different possible definitions and interpretations of musical voices. Cognitively, a voice simply refers to any set of notes which a listener may hear as a coherent melody, no matter whether that stream is monophonic (consisting of only non-overlapping notes) or polyphonic (containing any number of simultaneous notes). However, in certain styles of musical, the idea of a voice can even be ill-defined. For example, in some polyphonic piano music, voicing can be unclear from the score, and, even when voicing seems to be notated in the score, listeners may indeed disagree on which notes belong to each voice. Thus, in this chapter, we focus on music which has well-defined voices, such as fugues and quartets.

Additionally, from a computational standpoint, it can be useful to restrict the definition of a voice to strictly monophonic streams of notes, especially when performing voice separation or assignment as a preprocessing step for other music information retrieval (MIR) tasks. For example, existing work on rhythmic structure detection (van der Weij, 2012), pattern detection (Hsu, Liu, & Chen, 2001; de León & Inesta, 2007), query-by-tapping (Peters, Cukierman, Anthony, & Schwartz, 2006; Hanna & Robine, 2009), and query-by-humming (Birmingham, Dannenberg, & Pardo, 2006; Ryynanen & Klapuri, 2008) all require monophonic MIDI data as input. Furthermore, even among MIR techniques which can be run on polyphonic data, many still perform better when run on monophonic input. For example, Bruderer, McKinney, and Kohlrausch (2012) show that song segmentation is consistently more accurate when run on monophonic input. Additionally, voice separation could be useful as the first step towards a complete AMT system, and it was shown by Berg-Kirkpatrick et al. (2014) that the use of such a model improves the accuracy of music transcription significantly. Thus, for our work on voice separation, particularly in the context of creating a music language model for music transcription, we restrict ourselves to use only monophonic voices.

Towards the analysis of musical voices, Huron (2001) and Tymoczko (2008) investigate voice leading rules—rules which govern how voices evolve over time within a single piece—from a cognitive perspective, and offer valuable insights into possible voice separation rules, many of which we have applied to our model. Huron's *Common Tone Rule*, *Chordal Tone Rule*, and *Avoid Leaps Rule* all suggest that large melodic intervals between consecutive notes within a single voice should be avoided, a property which Tymoczko calls *efficient voice leading*. Likewise, Huron's *Part-Crossing Rule* and *Part Overlap Rule* suggest that two separate voices should not cross in pitch, even if one voice has fallen silent, a concept referred to by Tymoczko as *voice crossings*. A third principle suggested by Huron is that of temporal continuity—the idea that the stream of notes should be relatively continuous within a single voice, and not have too many gaps of silence. Temperley (2008) applies many of these concepts to constructing a successful probabilistic model of melodic perception, a task closely related to voice separation and assignment.

In Section 3.2, we introduce a voice separation model which is able to take as input polyphonic MIDI data, including live performance data, and separate it into strictly monophonic voices which can be used by MIR methods such as those listed above. Like other voice separation systems, it can be run as a preprocessing step to those existing methods, or as a standalone voice separation model on the data itself. Unlike most existing methods, however, our model allows for some limited overlap between consecutive notes within a single monophonic voice, and we show that using this feature, its accuracy remains high when separating live performance MIDI into voices. Next, in Section 3.3, we integrate this voice separation model with an acoustic multi-pitch detection model and use the joint system to perform both multi-pitch detection and voice assignment from audio recordings of a cappella music.

3.2 From Live Performance MIDI

This section presents an HMM which can be used to separate live performance MIDI into monophonic voices. It works on two basic principles: that consecutive notes within a single voice will tend to occur on similar pitches, and that there are short (if any) temporal gaps between them. We also present an incremental algorithm which can perform inference on the model efficiently, and show that our approach achieves state-of-the-art results when run on a corpus of 78 compositions by J. S. Bach, each of which has been separated into gold standard monophonic voices as suggested by the original score. We also show that it can be used to perform voice separation on live performance MIDI without an appreciable loss in accuracy. The code for the model described in this section is available at https://github.com/apmcleod/voice-splitting.

3.2.1 Related Work

There are many existing algorithms which perform voice separation, many of which are based on some of Huron and Tymoczyko's principles above, though not all of them were designed to be used for the same purpose as our model. Karydis, Nanopoulos, Papadopoulos, Cambouropoulos, and Manolopoulos (2007), for example, describe a more cognitively motivated voice separation model which attempts to separate MIDI data into polyphonic voices, something we avoid, as noted above. The approach first suggested by Kilian and Hoos (2002) and later expanded upon in a paper by Kilian (2004) has a large number of parameters which must be adjusted by the user at run time to produce the desired separation results, and therefore seems to be most useful as an aid to the manual transcription of MIDI data.

While the methods above cannot be applied directly to monophonic voice separation on live performance MIDI, that is not to say that all of the concepts and techniques used by them are irrelevant to the problem. Kilian (2004), for example, uses a Gaussian window function in their evaluation of voice continuity, something which we use as well in our model.

Chew and Wu (2004) propose a heuristic-based solution to the monophonic voice separation problem. Metronomic MIDI data are separated into chronological sections called "contigs," each of which represents a time period in a song during which there is a constant number of co-occurring notes. The notes within these contigs are then separated into monophonic sequences of notes called "fragments." Within a contig, no two fragments may cross in pitch. To join fragments from consecutive contigs into voices, a global optimisation approach is used which minimises the total pitch difference between consecutive notes in each resulting voice.

Ishigaki, Matsubara, and Saito (2011) introduce a new strategy for connecting contigs by prioritising those connections at locations where the number of co-occurring notes is increasing. This is based on the assumption that new voices are more likely to occur at a large pitch-distance away from the other voices than those which are just finishing, and thus should be easier to recognise. Guiomard-Kagan, Giraud, Groult, and Levé (2016) improve further on contig-based voice separation, using musical features

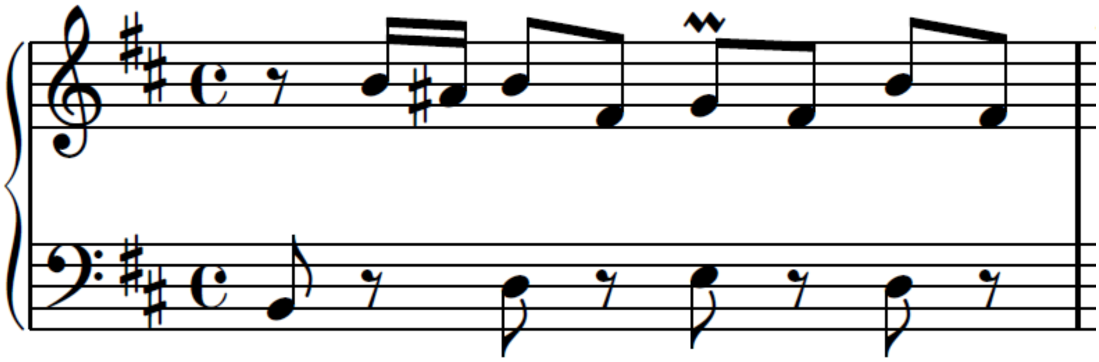


Figure 3.1: The first bar of the 15th Invention by Bach (BWV 786).

to compute a connection score for each potential fragment connection, and connecting fragments in a way that attempts to maximise the overall connection score. In this approach, however, note values are used during feature calculation, thus requiring metrically aligned data as input. Both of these approaches increase performance over the original contig approach of Chew and Wu (2004).

All of the contig approaches perform well in general, but they have two weaknesses. First, since they each perform a global optimisation, they cannot be run in real-time on live input. This may not seem like a problem when voice separation, a relatively computationally light task, is the only thing being computed on a song, but when it is only the first task in a series of other, more complicated MIR tasks, it could become more important. The second drawback is that they will always group a contig with *n* co-occurring notes into exactly *n* voices; however, there are cases where this is incorrect. For example, in Figure 3.1, an excerpt from Bach's 15th Invention (BWV 786), the first three notes will be grouped into a single monophonic contig, and then into a single voice, even though the proper separation would be to assign the first note to one voice, and group the second and third notes together in a different voice.

Moving past contigs towards statistical approaches, Kirlin and Utgoff (2005) describe VoiSe, a feature-based system for voice separation. However, it requires many features related to a song's time signature and tempo annotated, so it must be run on metronomic MIDI data rather than live performance data. Additionally, it is trained and evaluated on very limited excerpts (fewer than ten bars for each evaluation), and its performance lags behind that of Chew and Wu (2004).

Madsen and Widmer (2006) propose a solution based on pitch proximity which uses a small lookahead and tests all possible combination of grouping notes into voices within that lookahead, minimising a cost function. However, the completeness values they report in evaluation are substantially lower than the average voice consistency results reported by Chew and Wu (2004) when run on the same data. Madsen and Widmer themselves note that the two metrics are comparable (2006), so the difference between their scores cannot be written off as simply a difference between the evaluation metrics used.

Jordanous (2008) proposes a probabilistic approach to the problem of voice separation based roughly on Chew and Wu's method in that the input data is first searched globally for periods during which the voice structure is more obvious. Then, the surrounding notes are assigned to voices based on voice transition probabilities learned from the data. There are, however, two drawbacks to this method. First, similarly to Chew and Wu's, the global search cannot be performed in real-time on live input. Second, though the probabilities are based on pitch differences within a voice, temporal gaps between consecutive notes within a voice are disregarded. The results reported by Jordanous (2008) also fall below those of Chew and Wu (2004).

In recent years, neural models have been proposed for the task of voice separation. De Valk (2015) introduces three models—two neural networks and one HMM—designed originally for lute tablature, but also applicable to other forms of music. The neural models outperform the HMM significantly in all reported test cases. In practice, the neural models require being given a maximum number of voices a priori, although the HMM does not. All models are based on feature vectors for either notes or chords, containing information about both individual notes as well as the polyphonic context of each and metrical and rhythmic information (the metrical structure alignment must also be known a priori). The feature vectors are fed into the models, which produce a voice assignment for each note. Gray and Bunescu (2016) introduce another neural network model using a similar feature vector, though it requires only that the key signature of a piece is known a priori since it uses scale degree as a feature. It is also able to dynamically add new voices throughout the piece, thus not requiring as input the maximum number of voices.

None of the approaches discussed so far are able to be run directly on live performance MIDI without preprocessing the data with a quantisation step, which can add errors. Duane and Pardo (2009) propose a heuristic solution, which can be run on live performance data directly, where each note in a the input is treated as a node on graph, and edges are added to that graph grouping the corresponding notes into voices. Constraints are placed on which edges can be added to the graph so that the resulting voices are always monophonic. The program decides which edges to add based on a weight function dependent on each note pair's pitch and temporal difference. Nodes are first grouped into segments based on note onset times, and edges are added between nodes within each segment. Then, the segments themselves are tied together with edges based on another weighting function.

3.2.2 Proposed Solution

Our model, an HMM, is loosely based on Huron and Tymoczko's principles of pitch closeness and temporal continuity as described in Section 3.1 above. The model itself is presented in Section 3.2.2.1, and we present an algorithm which can be used to perform inference on our model incrementally in Section 3.2.2.2. A worked example of our model being run on some MIDI data is shown in Section 3.2.2.3.

One novel aspect of our model is that, to our knowledge, it is the first to investigate the principle of pitch closeness in a wider context than simply comparing the pitches of consecutive notes. Rather, we use a weighted average of the most recent notes in a voice (see Equation (3.4) below) and compare the pitch of a new note to that weighted average. This has the benefit of allowing some large jumps (which do occur, although they are rare) while still keeping each voice in a relatively stable pitch range, rather than immediately shifting each voice to the pitch of the most recent note.

Additionally, our solution has a few more advantages over some of the existing solutions. For one, it takes as input only note onset time, offset time, and pitch: no metrical or harmonic information is required. The maximum number of voices is also not required as input, nor is there a need to calculate it directly (many existing algorithms set this to the greatest number of concurrent notes in the song); rather, voices are added dynamically by the model as needed. Additionally, it allows for notes within a single voice to overlap slightly, which eliminates the need to perform any preprocessing such as quantisation on its input as most other solutions require. The fact that our model does not require any information or alignment a priori, and does not constrain voices to non-overlapping notes allows it to be run directly on live performances data, even if no information is known about the performed piece. The incrementality of our algorithm allows it in principle to be run in real-time, for example, on a live stream of data. Table 3.1 compares our model to the most successful existing approaches from Section 3.2.1 in the context of these properties.

Before getting into any more details, it will be useful to define the relevant properties of an input note n: pitch, onset time, and offset time. A note's pitch $\text{Pitch}(n)$ is

Model	Incremental	Live	Voices	Other
Chew and Wu (2004)			C	
Duane and Pardo (2009)		✓		
Ishigaki et al. (2011)			C	
de Valk (2015)			I	Metrical alignment
Gray and Bunescu (2016)	✓			Key signature
Guiomard-Kagan et al. (2016)			C	Metrical alignment
This work	✓	✓		

Table 3.1: A comparison of our proposed HMM to the most successful existing approaches from Section 3.2.1 in the context of four properties: (1) Incremental: whether the model is run incrementally from beginning to end; (2) Live: whether it can be applied directly to live performance MIDI data without quanitisation as a preprocessing step; (3) Voices: whether a maximum number of voices is required, either as input (I) or by an initial calculation (C) before voice separation; and (4) Other: what additional information (if any) is required a priori.

an integer on the range $[0-127]$ representing its pitch in equally-tempered keyboard semitones. A note's onset time $\text{On}(n)$ represents the number of microseconds between the beginning of the song and the onset of the note. Similarly, a note's offset time $\text{Off}(n)$, represents the number of microseconds between the beginning of the song and the offset of the note. It is also useful to define the duration of a note as the difference between its offset and onset times as in Equation (3.1).

$$\text{Dur}(n) = \text{Off}(n) - \text{On}(n) \tag{3.1}$$

3.2.2.1 Model

3.2.2.1.1 State Space Our model is an HMM where each state S represents a list of monophonic voices V_i. A voice V is a list of notes $n_{1\rightarrow n}$ ordered by onset time, where $\text{On}(n_i) < \text{On}(n_{i+1})$. Within a single state, no two voices may contain the same note. Furthermore, since we will apply our model to live performance data, rather than just metronomic MIDI data, we allow for some minimal overlap between consecutive notes within a voice to account for cases where the performer may remain on a note while beginning to play the next (a brief discussion of how often and to what magnitude this occurs in the data is found in Section 3.2.3.1). Specifically, we allow notes n_i and n_{i+1}

to overlap if and only if Equations (3.2) and (3.3) are both satisfied. Equation (3.2) ensures that the duration of the overlap comprises at most half of the duration of the first note involved, while Equation (3.3) ensures that the overlap does not continue for the entirety of the second note.

$$\mathrm{Off}(n_i) - \mathrm{On}(n_{i+1}) \leq \frac{\mathrm{Dur}(n_i)}{2} \tag{3.2}$$

$$\mathrm{Off}(n_i) < \mathrm{Off}(n_{i+1}) \tag{3.3}$$

Each voice V also has a pitch, calculated by the function given in Equation (3.4), where l is a tunable constant. A voice's pitch is simply a weighted average of its l most recent notes, where each successive note is weighed more heavily than the previous note by a factor of two. This allows the voices to gradually change in pitch over time, rather than jumping immediately to the pitch of each new note, while still enforcing Huron's *Avoid Leaps Rule*.

$$\mathrm{Pitch}(V) = \frac{\sum_{i=0}^{\min(l,|V|)} (2^i * \mathrm{Pitch}(n_{|V|-i}))}{\sum_{i=0}^{\min(l,|V|)} 2^i} \tag{3.4}$$

Since the onset and offset times of each note are unbounded, and there are likewise an unbounded number of notes in a given MIDI file, the state-space of our model is of infinite size. Thus, instead of using discrete emission and transition probabilities, we use emission and transition probability functions, described in the following sections.

3.2.2.1.2 Emission Function The emission function for each state is entirely deterministic: each state has exactly one possible emission with probability 1, although multiple states may have identical emissions. Each state outputs a set N of notes, where each note n within N has an equal onset time. That is, $\forall n, n' \in N, \mathrm{On}(n) = \mathrm{On}(n')$. Specifically, a state S outputs a set N of all notes n contained in any voice $V \in S$ which satisfy Equation (3.5). Conceptually, N is the set of the most recently played notes within the state, and $P(N|S) = 1$.

$$\mathrm{On}(n) = \mathrm{Max}(\mathrm{On}(n')), \forall n' \in V, \forall V \in S \tag{3.5}$$

3.2.2.1.3 Transition Function Before we define our transition probability function, we must define precisely what transitions exist within our model. A state S has a transition to state S' if and only if the following two conditions are satisfied: (1) the set of notes contained by any voice in S' which are not contained by any voice in S

must be exactly the set of notes determined by the emission function of S' as defined above; and (2) removing those notes from the voices of S', and then removing any voices from S' which become empty as a result, must produce exactly the voices of S, and these voices must appear in exactly the same order.

This transition from S to S' is represented by $T_{S,N,W}$, where S is the original state, N is a list of the notes from the emission function of S' ordered by increasing pitch, and W is a list of integers with exactly one element for each note $n_i \in N$, where w_i represents the voice $V \in S$ to which the corresponding note n_i should be added. No two w_i should be equal, and $\forall w_i, \text{abs}(w_i) \leq |S| + \text{Count}(w_i < 0)$. Each $T_{S,N,W}$ represents one individual note transition for each (n_i, w_i) pair. These note transitions are handled in order of increasing abs(w_i), and the value of w_i represents the following: if $w_i < 0$, add n_i to a new voice V inserted at the w_ith index of S; if $w_i > 0$, add n_i to the existing voice $V_{w_i} \in S$.

We can now define the probability of a given transition $P(S'|S) = P(T_{S,N,W})$ as simply the product of the probabilities of each individual note transition within it, as shown in Equation (3.6), multiplied by an order score which penalises a note being added to a voice out of pitch order.

$$\text{P}(T_{S,N,W}) = \prod_{0 \leq i \leq |N|} \text{P}(S, n_i, w_i) * \text{order}(S, n_i, w_i) \tag{3.6}$$

The order function by default returns 1, but that value is halved for each of the following cases that applies:

1. $|w| > 1$ and $\text{Pitch}(V_{|w|-1}) > \text{Pitch}(n)$

2. $0 < w < |S|$ and $\text{Pitch}(V_{w+1}) < \text{Pitch}(n)$

3. $-|S| \leq w < 0$ and $\text{Pitch}(V_{|w|}) < \text{Pitch}(n)$

Case 1 is applicable when a note will be added to a voice and the preceding voice in the state (if one exists) has a greater pitch than the pitch of the note, while cases 2 and 3 are mutually exclusive based on the sign of w, but each applies when a note will be added to a voice and the succeeding voice in the state (if one exists) has a lower pitch than the pitch of the note. Together, the cases of the order function work to minimise voice crossings, though such crossings are not disallowed completely.

The probability of each individual note transition is the product of its pitch score and its gap score, or a tunable constant if the note will be added to a new voice, as shown in Equation (3.7).

$$\mathrm{P}(S,n,w) = \begin{cases} \mathrm{pitch}(S,n,w) * \mathrm{gap}(S,n,w) & w > 0 \\ s_{new} & w < 0 \end{cases} \tag{3.7}$$

A note transition's pitch score is used to minimise melodic jumps within a voice, and is computed using the Gaussian window function as shown in Equations (3.8) and (3.9), where σ_p is a tunable parameter. A note transition's gap score is used to prefer temporal continuity within a voice, and is computed using the max function on the result of the logarithmic function, as shown in Equation (3.10), where σ_g and g_{min} are both tunable parameters.

$$\mathrm{pitch}(S,n,w) = \mathrm{Gauss}(\mathrm{Pitch}(n) - \mathrm{Pitch}(V_w), \sigma_p) \tag{3.8}$$

$$\mathrm{Gauss}(\mu,\sigma) = e^{-\frac{1}{2}(\frac{\mu}{\sigma})^2} \tag{3.9}$$

$$\mathrm{gap}(S,n,w) = \max\left(\ln\big(-\frac{\mathrm{On}(n) - \mathrm{Off}(\mathrm{last}(V_w))}{\sigma_g} + 1\big) + 1, g_{min}\right) \tag{3.10}$$

Note that the offset time of the note within a voice is not used in any of the above equations, so any two notes with equal pitch and onset time are treated as equally likely by our model, regardless of their offset times (given that each transition will create a valid resulting state). Note also that the transition probabilities out of a given state do not necessarily sum to 1, but this is not important, as the values can be normalised with a constant factor.

3.2.2.2 Inference

To find the most likely final state given our observed note sets we use a slightly modified Viterbi algorithm. (See Viterbi (1967) for an overview of the algorithm.) Our observed data are the notes found in a given MIDI file, ordered by onset time. If multiple notes have equal onset time, they are observed as a set.

We made two modifications to the Viterbi algorithm, each related to reducing the search space of the algorithm, since the size of the search space increases exponentially without any constraints. The first modification is applied when adding a note into a new voice. Specifically, we only check those voice indices w which have the maximum order score of any valid index. In practice, this ensures that we only try to insert a new voice in pitch order with the surrounding voices when those voices have not crossed. (When any surrounding voices have crossed, we try to insert the new voice at every index bordering those voices). Secondly, we use beam search with a tunable beam size

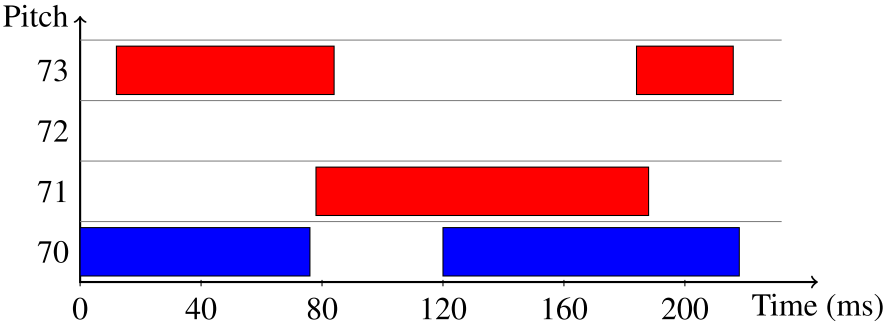


Figure 3.2: An example of the notes that might be found in a MIDI file, displayed in a piano roll format. Here, each note is colour-coded based on the voice to which our HMM would assign it.

b. After each iteration of the algorithm, we save only the b most likely states given the observed data to that point.

3.2.2.3 Example

To help illustrate the workings of the model, we will now go through an example of the Viterbi algorithm being run on it, given the MIDI data shown in Figure 3.2 (and assuming some reasonable setting of the parameters). Here, the notes have been colour-coded according to the voice to which each will be assigned by our model. For a diagram of this example, see Figure 3.3 where each note n is represented as $[\text{Pitch}(n), \text{On}(n)]$, and voices within a state hypothesis are grouped using braces. For simplicity, we use a beam size of 2 in this example.

The initial state, S_0, is empty, as no notes have been observed yet. After seeing N_1, there is no decision to be made, as the only valid transition is to add the observed note into a new voice. Upon observing N_2, we again have only one valid transition to check. We cannot add this new note into the existing voice because it fails the overlap constraint in Equation (3.2), and the new voice must be placed at index 2 of our state due to it being out of pitch order with the existing V_1, because of the optimisation mentioned in Section 3.2.2.2.

Once we observe N_3, however, there are a few possible state transitions to check. Trivially, we could add the new note into a new voice at index 2 (the other indices would be out of pitch order again), but assuming that s_{new} has been set sufficiently low, this transition will not be saved due to our beam size of 2. The new note could also be added to either of the existing voices. Both the pitch score from Equation (3.8) and the

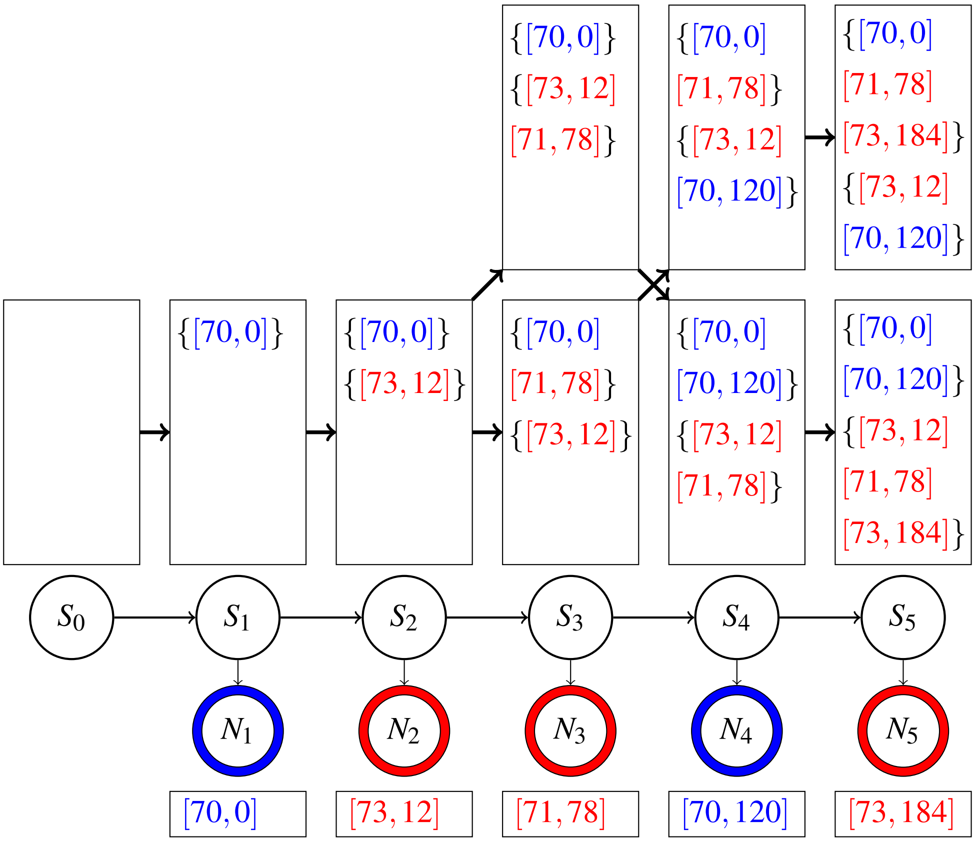


Figure 3.3: An example of our model being run on the notes from Figure 3.2 with a beam size of 2. Each observed note set's border, and each note, is colour-coded based on the voice to which it is finally assigned. The two most likely state hypotheses at each step are listed in the large rectangles above the state diagram, with the more likely hypothesis appearing on the bottom row. Each state hypothesis has an incoming arrow indicating which prior state hypothesis was used to transition into that state.

gap score from Equation (3.10) are slightly greater when adding the new note to V_2, so we assign that transition a higher probability.

Next, we observe N_4, and out of each current hypothesis state, besides adding any new voices, the only valid transition is that which adds the new note into the voice which does not currently contain the note $[71,78]$, due to Equation (3.2). The difference in probability between the two remaining transitions is about a factor of 2, since performing the given transition on the current most likely hypothesis state introduces a pitch crossing, and therefore has an order score of $1/2$ in Equation (3.6), due to case 1. Therefore, the two current hypothesis states each perform the transition and then switch in order, as indicated by the arrows in the diagram.

Finally, we observe N_5, and out of each current hypothesis state, besides adding any new voices, the only valid transition is that which adds the new note into the voice which does not currently contain the note $[70,120]$, due to Equation (3.3). So, we perform the valid transition on each hypothesis state, and the orders will not change due to the transition probabilities being relatively close to each other.

3.2.3 Evaluation

3.2.3.1 Corpora

We evaluate our model on six distinct corpora:

1. The 15 two-part inventions by J. S. Bach.[1]

2. The 15 three-part sinfonias by J. S. Bach.[1]

3. The 24 fugues from *The Well-Tempered Clavier, Book 1 (WTC I)* by J. S. Bach.[2]

4. The 24 fugues from *The Well-Tempered Clavier, Book 2 (WTC II)* by J. S. Bach.[2]

5. The 28 movements from *String Quartets*, Op. 1, by J. Haydn.[3]

6. The 19 live performances of J. S. Bach inventions (5), fugues (5), and preludes (9) (from *WTC I* and *WTC II*) from CrestMusePEDB, introduced by Hashida, Matsui, and Katayose (2008).

[1] The inventions and the sinfonias were acquired from `www.imslp.org`.

[2] The fugues were acquired from `www.musedata.org`.

[3] The quartets were acquired from `www.kunstderfuge.com`.

The first five corpora contain only metronomic MIDI data, while the sixth consists solely of live performance MIDI data. For the Bach compositions (datasets 1–4 and 6), each gold standard voice corresponds to an individual fugal voice as suggested by the original score. For the Haydn quartets, each separate instrument part (violin I, violin II, viola, and cello) is used as a gold standard voice, as in Duane and Pardo (2009); however, these gold standard voices may not be entirely correct cognitively (as they also note), since the melody often switches between instruments, especially between the two violin parts.

In rare cases, most often on the final note of a piece, a single voice may contain a chord. This is a problem for our model since we separate the pieces into strictly monophonic voices. Therefore, in such cases, we manually remove all but the lowest-pitched note in the chord as a preprocessing step. This makes musical sense, since it has been suggested by Dixon (2001) among others that notes with lower pitch are more salient than those with higher pitch. This sort of preprocessing is quite common: all models which separate notes into monophonic voices must handle chords in an ad hoc fashion, and those which we compare against either remove individual notes as we do, or remove all sections of each piece which contain any chords. Such preprocessing is particularly important for models which calculate the number of voices as the greatest number of concurrent notes in each piece. In practice, since our model adds new voices dynamically, running it without this preprocessing tends to result in it simply adding an additional voice or two containing only the extra chord notes, particularly for those pieces in which chords are only present at the end (such as those found in our corpora). For pieces with more frequent chords, however, we would expect our model's performance to drop, and some additional logic would need to be added to handle such chordal voices.

Each dataset presents a slightly different challenge. The inventions are the simplest of the compositions, each containing exactly two voices. The sinfonias are slightly more complicated, containing three voices each, and the fugues are the most complicated of the Bach compositions, sometimes even containing more than four voices. The quartets each consist of exactly four parts, but they can be more complicated than the fugues in that the different parts, especially the two violin parts, are liable to cross in pitch during a piece. The live performances are difficult in that the data is not clean. That is, note onsets which, in the score, occur immediately after another note's offset, may not occur precisely at that time. This adds some noise into the data and makes voice separation more difficult.

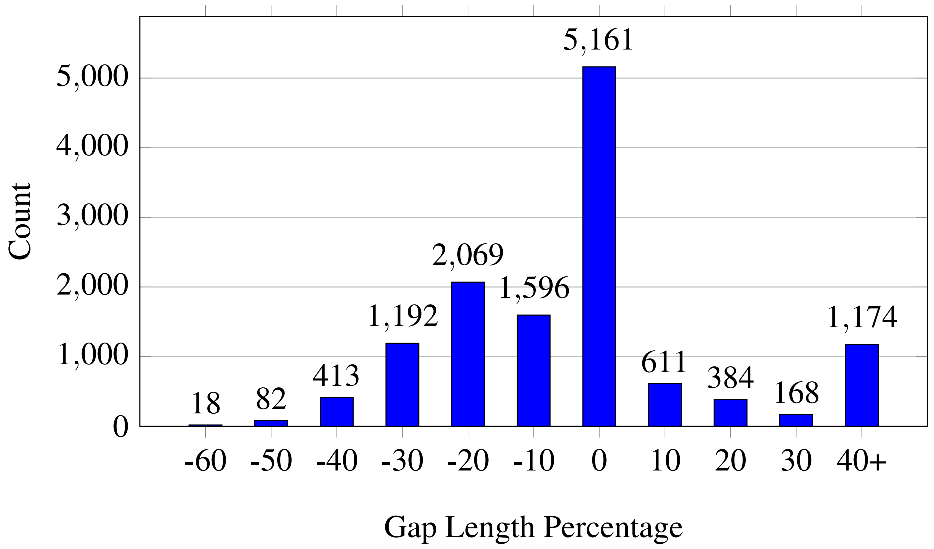


Figure 3.4: A histogram of the gap lengths between consecutive notes in the live performances, omitting those separated by a rest in the score, normalised as a percentage of the initial note's duration. Each bucket contains those percentages within 5% of the bucket label, with the exception of the bucket labelled "40+", which contains all gap length percentages greater than 35%. A negative value indicates an overlap between notes, while a positive value indicates a gap.

The extent of this live performance noise is quantified by the histogram in Figure 3.4. We measure the gap length between each pair of consecutive notes within every voice of the live performances, omitting those pairs which are separated by a rest in the score, resulting in a total of 12868 note pairs. We then normalise the gap lengths by the duration of the initial note, and separate the resulting gap length percentages into the buckets shown. Each bucket contains those percentages within 5% of the bucket label, with the exception of the bucket labelled "40+", which contains all gap length percentages greater than 35%. A negative value indicates an overlap between notes, while a positive value indicates a gap. It can be seen that, while about 40% of consecutive note pairs have only a minimal overlap or gap, greater than 10% of them overlap by at least one quarter of the initial note's duration, and an additional 10% are separated by a gap of at least that length. Note that performing the same calculation on any of the other corpora would result in every note pair having a gap length of exactly 0.

3.2.3.2 Baseline Methods

We compare our model to the best scoring models from Section 3.2.1, as summarised in Table 3.1. Only one of the models (Duane & Pardo, 2009) can be run directly on live performance input, and most of the others require the data to be more heavily annotated than our model does. Thus, for comparison with Duane and Pardo (2009), we have run their code directly on our data and report results on every dataset. However, for the other methods, we simply compare against the numbers reported in their respective papers, on whichever of our datasets they also use. For de Valk (2015), we use the highest scoring model in all cases, in application mode (as opposed to test mode, in which the model uses ground truth voice assignment—rather than its own voice assignments—as features for subsequent voice assignments). For Guiomard-Kagan et al. (2016), we use the GA1 model, again the highest scoring.

3.2.3.3 Metrics

We use two different evaluation metrics: average voice consistency (AVC), as introduced by Chew and Wu (2004); and F-measure, as used by Duane and Pardo (2009). Statistical significance is calculated using a two-tailed t-test.

Before we can explain AVC, a couple of definitions are needed. First, let $\text{voice}(n)$ be the ground truth voice to which a note n belongs. Second, let $\text{voice}(V)$ be the voice to which the majority of notes $n \in V$ belong. Then the Voice Consistency of a voice $\text{VC}(V)$ is given by Equation (3.11).

$$\text{VC}(V) = \frac{|\{n \in V : \text{voice}(n) = \text{voice}(V)\}|}{|V|} \tag{3.11}$$

The AVC of a given voice separation hypothesis state S is simply an average of the voice consistencies (VCs) of every voice $V \in S$ as shown in Equation (3.12). A problem can be seen with this metric, particularly for a model such as ours which is able to add new voices dynamically. Specifically, if our model puts a single note into a voice by itself, the VC of that voice is a perfect 1.0. Extrapolating further, if a model were to put every single note of a piece into a separate voice, it would receive a perfect AVC of 100. Thus, we do not use AVC as our main evaluation metric, rather only for comparison against Chew and Wu (2004) and Ishigaki et al. (2011) (on those datasets which they also use), because we were unable to get either of their implementations.

Parameter	Min	Max	Min Step
l	1	12	1
s_{new}	1×10^{-11}	1×10^{-7}	0
σ_p	3	9	0.5
g_{min}	1×10^{-6}	0.1	0
σ_g	1×10^{4}	1×10^{6}	0

Table 3.2: The minimum, maximum (both inclusive), and minimum step settings used for each parameter during our grid search.

$$\text{AVC}(S) = 100 * \frac{\sum\limits_{V \in S} \text{VC}(V)}{|S|} \tag{3.12}$$

We report F-measure values for all of our corpora, and use it for our main evaluation. The F-measure we use is just the standard F-measure, where we treat the voice separation problem as one of binary classification where between each pair of notes, our model must decide whether the two notes occur consecutively within a single voice or not. Then, the F-measure is calculated by Equation (3.13).[4]

$$F\text{-}measure = 2\frac{precision * recall}{precision + recall} \tag{3.13}$$

3.2.3.4 Training

To train our model, we used a grid search. For each of the parameters, we manually set a minimum and a maximum value, both inclusive. For some, we also set a minimum step size, limiting the number of values to check even if the grid size we picked was very small. See Table 3.2 for these values for each of our model's parameters. For all training, our beam size is restricted to 10. This is done to speed up training time, though during testing we use a beam size of 25.

We use different training set splits for each of our test sets to avoid overfitting. When evaluating on the inventions, we trained on the sinfonias and the fugues from both books one and two. When evaluating on the sinfonias, we trained on the inventions and the fugues from both books one and two. When evaluating on the fugues from books one and two, we trained on the inventions and the sinfonias. When evaluating on the Haydn quartets, we used leave-one-out cross validation between the six

[4] Some papers report recall and precision values as *soundness* and *completeness* respectively.

Corpus	l	s_{new}	σ_p	g_{min}	σ_g
Inventions	6	1×10^{-9}	4	8×10^{-4}	127000
Inventions*	9	3×10^{-8}	6	9×10^{-5}	127000
Sinfonias	5	4×10^{-8}	4	9×10^{-5}	127000
Sinfonias*	9	2×10^{-8}	6	6×10^{-5}	127000
WTC I & II	11	5×10^{-10}	4	7×10^{-5}	224000
WTC I & II*	9	1×10^{-9}	5.5	8×10^{-5}	224000
Haydn 1	9	1×10^{-10}	4	9×10^{-5}	321000
Haydn 2	8	2×10^{-11}	5	8×10^{-5}	321000
Haydn 3	7	2×10^{-8}	7.5	0.01	20000
Haydn 4	7	1×10^{-9}	4	8×10^{-5}	321000
Haydn 5	9	1×10^{-9}	4	7×10^{-5}	321000
Haydn 6	7	1×10^{-11}	4	8×10^{-5}	321000
Live Inventions	7	1×10^{-9}	4	0.01	515000
Live WTC	6	3×10^{-8}	6	0.01	806000

Table 3.3: The parameter settings used when evaluating our model on our different test sets. A * denotes where we have trained to optimise AVC rather than F-measure.

different quartets within the corpus. That is, for each of the six quartets, we trained on the other five when evaluating on the sixth. When evaluating on the live corpus from the CrestMusePEDB, we used leave-one-out cross validation between the inventions and the fugues and preludes. That is, when evaluating the inventions, we trained on the fugues and preludes and vice versa. The parameter settings used when evaluating each set corpus are shown in Table 3.3.

Finally, to investigate our model's ability to generalise across different musical styles, we also report its performance on WTC I & II using the settings originally used for Haydn 1, as well as its performance on the Haydn quartets using the settings originally used for WTC I & II.

3.2.3.5 Results

First, we present our AVC results on Bach's inventions, sinfonias, and fugues from WTC I & II, and compare them to those reported by Chew and Wu (2004) and Ishigaki et al. (2011). Although we do not believe that AVC is the best metric (see Section

Corpus	Chew and Wu (2004)	Ishigaki et al. (2011)	This Work
Inventions	99.29	98.73	**99.30**
Sinfonias	93.35	**95.27**	94.30
WTC I & II	84.39	**89.21**	88.23
Overall	88.98	**92.21**	91.53

Table 3.4: A comparison of Average Voice Consistencies between our work and those reported by Chew and Wu (2004) and Ishigaki et al. (2011).

3.2.3.3), we use it here for three reasons: (1) Both Chew and Wu (2004) and Ishigaki et al. (2011) report AVC results, and we do not have their implementations; (2) it is the only metric reported by both; and (3) neither reports F-measure, our main metric. They each evaluate on the same pieces, though they handle chords by not separating the last few notes of each input piece. Still, the results should be comparable, and they are shown in Table 3.4. It is important to note that the scores found in this table are averaged over all pieces in each category. That is, a value of 99.29 on the Inventions means that the mean AVC over all 15 Inventions was 99.29.

Our model sees an improvement over Chew and Wu's results, especially in the fugues. Both our improvement there and our overall improvement are statistically significant ($p < .05$). We don't see much improvement on the inventions or the sinfonias, most likely due to those pieces being simpler than the fugues. With only two or three parts, there are fewer mistakes to be made, and therefore, there is much less room for improvement. The difference in performance between our model and that of Ishigaki et al. (2011) is not statistically significant.

Here, we look more closely at one simple case where our model outperforms both Chew and Wu's as well as Ishigaki et al.'s, specifically the 15th invention (BWV 786), which was mentioned in Section 3.2.1 above. The first bar's piano roll notation is reproduced here in Figure 3.5. The red notes are all part of one voice, and the blue notes are all part of a second voice. Both our model and the other approaches program get this correct. However, while their programs incorrectly group the yellow note with the red notes, our model is able to correctly group it with the blue notes. This is entirely due to the contig constraint that portions of a song with the same number of simultaneous notes (in this case one), must be grouped into exactly that many voices. In fact, the model of Guiomard-Kagan et al. (2016) also makes this error since it is also based on contigs.

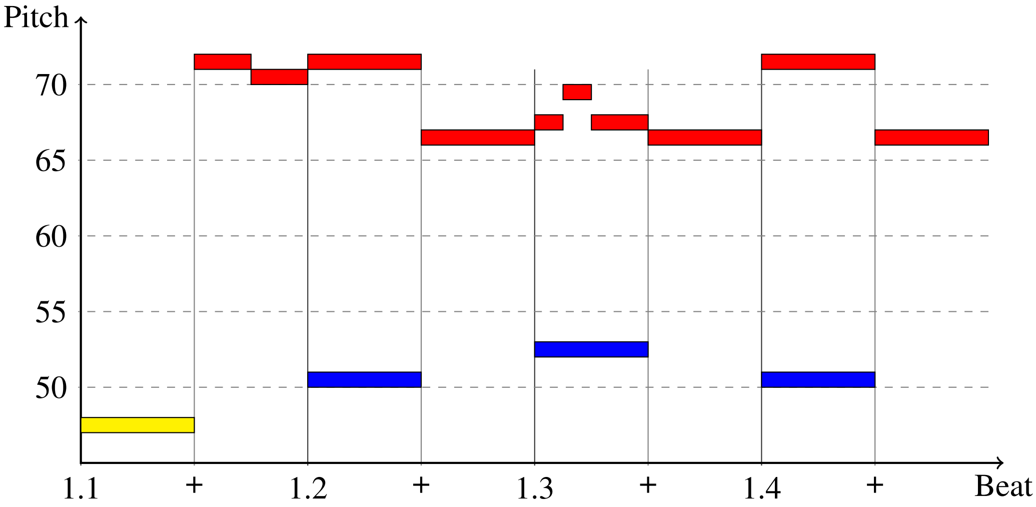


Figure 3.5: Invention 15 (BWV 786) in piano roll notation. The red notes all belong to one voice, and the blue notes another. The yellow note is the one which would be incorrectly grouped with the red notes by any contig-based approach (Chew & Wu, 2004; Ishigaki et al., 2011; Guiomard-Kagan et al., 2016), though our model groups it correctly with the blue notes.

Next, we present our model's F-measure results, and compare them against those from Duane and Pardo's (2009) program (the highest scoring of those mentioned in Section 3.2.1 which can handle live performance input) on our data. Shown in Table 3.5, we see definite improvement over their results on the Bach compositions, significantly for the sinfonias and all of the fugues ($p < .05$). On the Haydn quartets, however, our model performs slightly (though not significantly) worse than theirs, though we believe this to be due to the problem with the gold standard as mentioned in Section 3.2.3.1 above. On the live Bach performances, we again see significant improvement ($p < .05$), this time on both the inventions and the WTC performances.

The table also presents the performance of our model with the same parameter settings, except the pitch history length l set to 1, effectively removing the pitch history. This forces our model to check for pitch closeness between consecutive notes only (as is done in previous work), rather than using the running average of the most recent pitches in a voice (a novel aspect of our the proposed model). With l set to 1, the F-measure for each corpus either remains the same, or drops by 0.01, which is not a statistically significant difference. However, since large jumps in pitch within a voice are quite rare and such jumps are precisely the cases we would expect the pitch history to improve performance, it would be unreasonable to expect a statistically significant

Corpus	Duane and Pardo (2009)	This Work	This Work ($l = 1$)
Inventions	0.98	**0.99**	**0.99**
Sinfonias	0.91	**0.97**	**0.97**
WTC I	0.92	**0.97**	0.96
WTC II	0.91	**0.96**	**0.96**
WTC I & II	0.91	**0.97**	0.96
Haydn	**0.80**	0.79	0.79
Live Inventions	0.91	**0.98**	0.97
Live WTC	0.82	**0.94**	0.93

Table 3.5: A comparison of the F-measures achieved by the program described by Duane and Pardo (2009) against those of two versions of our model: one standard version, and another using the same parameter settings, except with the pitch history length l set to 1.

difference. Nonetheless, the fact that we do see a consistent decrease in performance without the pitch history, small though it is, does suggest that the use of a pitch history adds value to our model.

Investigating our model's ability to generalise across different musical styles, we also examine our its performance on WTC I & II using the settings originally used for Haydn 1, as well as its performance on the Haydn quartets using the settings originally used for WTC I & II. Both tests result in F-measures not significantly different from our original reported F-measures (0.96 and 0.79 respectively), showing that our model can indeed generalise across varied musical styles.

Diving more deeply into the comparison to Duane and Pardo's model, it appears that much of our improvement has come because our model more aggressively joins notes together into voices. That is, most of the errors that their program has which ours corrects are false negatives on their part. This occurs most often when a voice contains a rest, and is therefore absent from a piece for a beat or more. For example, in bars 18 and 19 of the first fugue from the Well-Tempered Clavier (BWV 846), one of the voices rests for two beats. This is shown in Figure 3.6, where the bold yellow notes are those which our model correctly joins, but Duane and Pardo's program fails to.

We also compare our model's results against those of de Valk (2015), Gray and Bunescu (2016), and Guiomard-Kagan et al. (2016), although each of these models require much more information a priori than our model does (see Table 3.1). De Valk

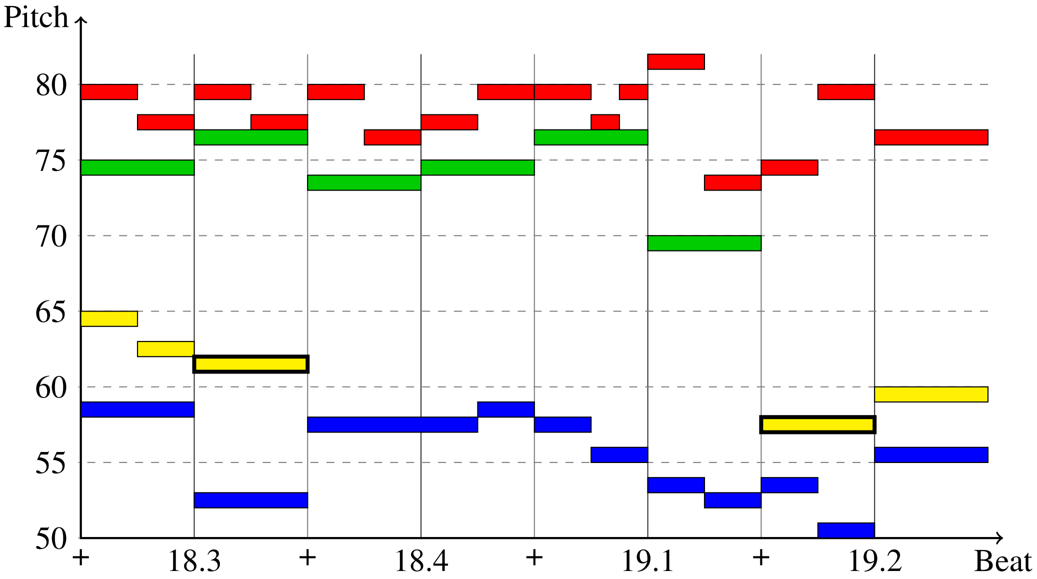


Figure 3.6: A piano roll representation of a portion of bars 18 and 19 from the 1st fugue in the Well-Tempered Clavier (BWV 846). Each colour represents a different voice. The two bold yellow notes are those which Duane and Pardo's program does not join into a single voice, though our model does.

(2015) and Gray and Bunescu (2016) each report results on a combined Bach corpus consisting of the Inventions, Sinfonias, and WTC I & II. Their F-measures of 0.96 and 0.95 respectively fall just short of our model's F-measure of 0.97 over the same set of pieces (though neither of these differences are statistically significant). Guiomard-Kagan et al. (2016) use WTC I as a training set, and report test results on WTC II, achieving an F-measure of 0.97, slightly higher than our F-measure of 0.96, though this difference is again not statistically significant. From these comparisons, it is clear that, although each of these three models requires a priori information or alignment about for each input file and cannot be run directly on live performance data, none of them offer a statistically better voice separation than our model, which requires none.

An example where our model could find an improvement is in the 8th bar of the fugue from the 41st fugue in the Well-Tempered Clavier (BWV 886). A piano roll representation of that bar, in which the correct voices have been colour-coded, is shown in Figure 3.7. The difficulty here is that the highest voice (red) ends at the exact same time that the lowest voice (blue) begins. Rather than starting a new voice, our model incorrectly shifts each existing voice down one as shown by the arrows in the figure. One thing which might help our model in separating the voices properly in cases such as this is looking for repeated patterns in the music. In the piece from this

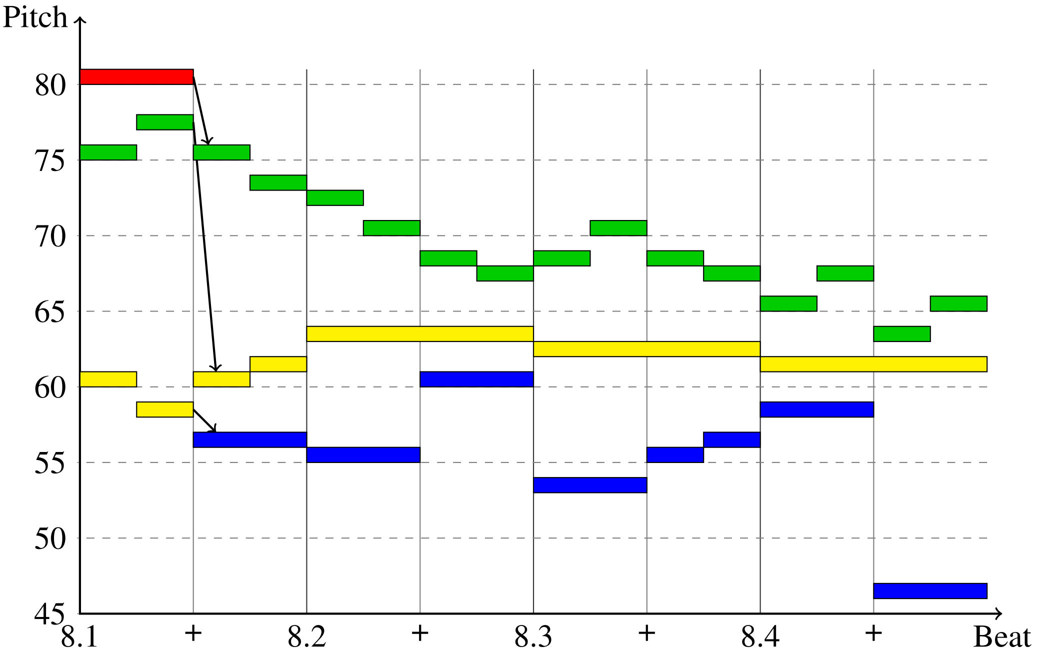


Figure 3.7: A piano roll representation of the beginning of bar 8 from the 41st fugue in the Well-Tempered Clavier (BWV 886). Each colour represents a different voice, and the arrows represent mistakes made by our model.

example, each time a new voice enters, it plays nearly the same pattern of notes (both rhythmically and melodically) for two bars. A model which is able to detect such patterns would be able to recognise that pattern occurring in the lowest voice and infer that those notes likely belong to a new voice. Additionally, the second-to-lowest voice (yellow) is still playing the tail end of that pattern, and thus should probably continue the pattern as the yellow notes in the figure do.

Another example of when our model errs is when two voices cross. Such cases are difficult in general, given that the tendency of voices is not to cross; however, there are often enough rhythmic or harmonic clues to allow a model to detect such crossings. For example, in the 73rd bar of the fourth fugue in the Well-Tempered Clavier (BWV 849), the two middle voices cross. (The two voices in question are reproduced, colour-coded, in Figure 3.8). The red voice contains only half notes, while the blue voice contains only 8th notes. A model which is able to take such rhythmic information into account should be able to detect the correct voice separation in this case; however, as we want to avoid requiring any metrical alignment as input, such rhythmic information would have to come from the use a beat tracking model.

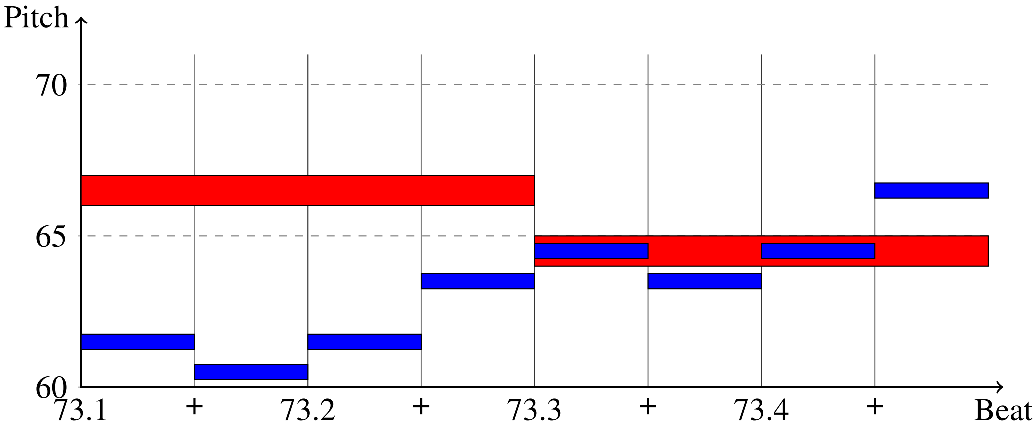


Figure 3.8: A piano roll representation of the middle two voices, colour-coded, during the 73rd bar of the fourth fugue in the Well-Tempered Clavier (BWV 849). Here, our model does not detect that the two voices have crossed even though this should be clear based on the rhythms of each of the voices. (The red notes are thicker here only so the crossing is easier to see.)

3.2.4 Conclusion

In this section, we have presented a new model for separating polyphonic MIDI data into a set of monophonic voices, and argued that this sort of model can play a central role in other MIR tasks. We have shown that our model achieves state-of-the-art results on a variety of music, including Inventions, Sinfonias, and Fugues by Bach, as well as Haydn String Quartets, and additionally, that it still achieves state-of-the-art results when run on live performance data.

One advantage of our model is that it takes as input only note onset time, offset time, and pitch; no metrical or harmonic information is required, unlike in many existing voice separation approaches. The maximum number of voices is also not required as input, nor is there a need to calculate it directly (many existing programs set this to the greatest number of concurrent notes in the song). Rather, voices are added dynamically by the model as needed. Our model can also be applied directly to live performance data, without requiring note quantisation as a preprocessing step (and thus introducing possible errors) like almost all existing approaches. Finally, our model can be evaluated incrementally, and can therefore be used for real-time applications, starting processing before the entire piece is available. This combination of properties makes our model unique among existing approaches, and yet it still achieves state-of-the-art results, even compared with those models which require much more information a priori.

An additional novel feature of our model is its use of a pitch history when checking for large jumps in pitch within a voice. This effectively allows for some jumps to occur while still keeping each voice within a relatively stable pitch range. That the use of this pitch history seems to increase performance suggests that it is an important aspect of voice separation, and should be included in future models.

A shortcoming of the model is that it is easily confused by long-range dependencies. For example, if there is a long rest in one or more voices, our model sometimes has trouble deciding where to assign the notes which follow the rest. It was shown by Granroth-Wilding and Steedman (2014), that long-range harmonic dependencies do exist in music, and can be parsed successfully, so such long-range voice connections should be possible to identify. One thing which might solve this and other mistakes which our model makes would again be to incorporate some knowledge of rhythmic, melodic, and harmonic patterns into it, as mentioned in relation to Figures 3.7 and 3.8 above. Given that we want to avoid the use of any a priori information or alignment, this would require us to combine the model with some sort of beat tracker, for example the one in (Dixon, 2001), in addition to a pattern detection algorithm, such as the one proposed by Hsu et al. (2001), recalculating our model's transition probabilities by increasing the probabilities of those transitions which would continue some pattern.

Future work could also update our model to use learned transition probability distributions rather than our somewhat naive Gaussian window and log score functions. We were unable to learn transition probabilities because of our small data set and large state space, but with a larger corpus of data, we will be able to apply machine learning to more closely approximate the true probability distributions of transitions with different pitch and gap differences, which may significantly improve our model's performance.

3.3 From Audio

This section presents a system for multi-pitch detection and voice assignment of audio recordings of a cappella performances (where the number of voices is known a priori) consisting of an integrated acoustic model and music language model. The acoustic model, which performs spectrogram decomposition, extends probabilistic latent component analysis (PLCA) using a 6-dimensional dictionary with pre-extracted log-spectral templates, while the music language model is a modified version of the voice separation HMM presented in the previous section. By integrating the two models, the system is able to detect multiple concurrent pitches in polyphonic vocal recordings and assign each to a specific voice type of soprano, alto, tenor or bass (SATB). We compare our system against multiple baselines, achieving state-of-the-art results for both multi-pitch detection and voice assignment on a dataset of Bach Chorales and another of Barbershop Quartets. We also present an additional evaluation of our system using varied pitch tolerance levels to investigate its performance at 20-cent pitch resolution. The code for the system described here is available at http://inf.ufrgs.br/~rschramm/projects/music/musingers.

To our knowledge, this is the first attempt to integrate an acoustic model with a music language model for the task of voice or instrument assignment from audio, as well as the first attempt to propose a system for voice assignment in polyphonic *a cappella* music. The approach described in this section focuses on recordings of singing performances by vocal quartets without instrumental accompaniment; to that end we use two datasets containing *a cappella* recordings of Bach Chorales and Barbershop quartets. The proposed system is evaluated both in terms of multi-pitch detection and voice assignment, where it reaches an F-measure of over 70% and 50% for the two respective tasks.

3.3.1 Related Work

This section presents related work on multi-pitch detection with a specific focus on vocal music and voice separation. An overview of prior work on multi-pitch detection in general can be found in Chapter 2, while a discussion of related work on MIDI voice separation can be found in Section 3.2.1.

In the context of multi-pitch detection, vocal music has been less often studied than instrumental music, likely due to the complexity and variety of sounds which can be produced by a singer. The timbre of two singers' voices can differ greatly, and even for a single singer, different vowel sounds produce extremely varied overtone patterns. Bohak and Marolt (2016) propose a method for transcribing folk music containing both instruments and vocals which takes advantage of melodic repetitions present in such music using a musicological model for note-based transcription. A less explored type of music is *a cappella*; in particular, vocal quartets constitute a traditional form of Western music, typically dividing a piece into multiple vocal parts such as soprano, alto, tenor, and bass (SATB). Schramm and Benetos (2017) propose an acoustic model

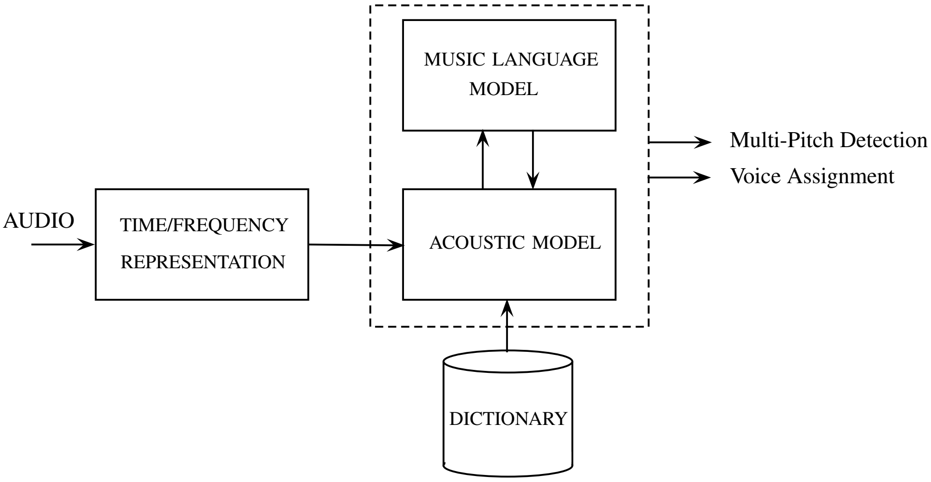


Figure 3.9: Proposed system diagram.

based on spectrogram factorisation for multi-pitch detection of such vocal quartets (a modified version of which we use here for our acoustic model).

A small group of systems (Bay, Ehmann, Beauchamp, Smaragdis, & Downie, 2012; Duan, Han, & Pardo, 2014; Grindlay & Ellis, 2011) have attempted to go beyond multi-pitch detection, towards *instrument assignment*—also called *timbre tracking*—where systems detect multiple pitches and assign each pitch to a specific source that produced it. Bay et al. (2012), for example, track individual instruments in polyphonic instrumental music using a spectrogram factorisation approach with continuity constraints controlled by an HMM. However, to our knowledge, no methods have yet been proposed to perform both multi-pitch detection and voice assignment from polyphonic vocal music.

3.3.2 Proposed Method

In this section, we present a system for multi-pitch detection and voice assignment applied to audio recordings of polyphonic vocal music (where the number of voices is known a priori) that integrates an acoustic model with a music language model. First, we describe the acoustic model, a spectrogram factorisation process based on PLCA. Then, we present the music language model, a modified version of the HMM described in Section 3.2. Finally, a procedure is proposed for the integration of these two components. Figure 3.9 illustrates the proposed system pipeline.

3.3.2.1 Acoustic Model

The acoustic model is a variant of the spectrogram factorisation-based model proposed by Schramm and Benetos (2017). The model's primary goal is to explore the factorisation of an input log-frequency spectrogram into components that have a close connection with singing characteristics such as voice type and the vocalisation of different vowel sounds. We formulate the model dictionary templates into a 6-dimensional tensor, representing log-frequency index (from the spectrogram's log-frequency axis), singer source (out of a collection of singer subjects used to construct the input dictionary), pitch (in equally-tempered semitone scale), tuning deviation (from perfect equal temperament) with 20-cent resolution, vowel type (English pure vowels), and voice type (bass, baritone, tenor, alto, or soprano). Similar to Grindlay and Ellis (2011), the singer source and vowel type parameters constrain the search space into a mixture-of-subspaces, clustering a large variety of singers into a small number of categories. In our model, the voice type parameter corresponds to the vocal part (SATB), where each vocal part is linked to a distinct set of singers (the singer source). For details on the dictionary construction, see Section 3.3.2.1.2. As time-frequency representation we use a normalised variable-Q transform (VQT) spectrogram[5] (Schörkhuber, Klapuri, Holighaus, & Dörfler, 2014) with a hop size of 20 ms and 20-cent frequency resolution. For convenience, we have chosen a pitch resolution that produces an integer number of bins per semitone (five in this case) and is also close to the range of just noticeable differences in musical intervals (Benetos & Holzapfel, 2015).

The input VQT spectrogram is denoted as $X_{\omega,t} \in \mathbb{R}^{\Omega \times T}$, where ω represents log-frequency and t time. In the model, $X_{\omega,t}$ is approximated by a bivariate probability distribution $P(\omega,t)$, which is in turn decomposed as in Equation (3.14).

$$P(\omega,t) = P(t) \sum_{s,p,f,o,v} P(\omega|s,p,f,o,v)P_t(s|p)P_t(f|p)P_t(o|p)P(v)P_t(p|v) \qquad (3.14)$$

$P(t)$ is the energy of the spectrogram at time t (a known quantity, calculated as $\sum_t P(\omega,t)$), and $P(\omega|s,p,f,o,v)$ is the fixed, pre-extracted spectral template dictionary (see Section 3.3.2.1.2). The variable s denotes the singer index (out of the collection of singer subjects used to construct the input dictionary), $p \in \{21,...,108\}$ denotes pitch

[5] The VQT spectrogram uses a linearly widening bandwidth (Q) towards the lower frequencies, resulting in a significantly simpler computation that requires a much shorter window length. This variable bandwidth closely models human perception, and the VQT can therefore be used for music without appreciable loss in accuracy because composers, knowing this, tend not to write music with closely-spaced pitches at lower frequencies.

in equally-tempered semitone resolution (in MIDI scale), f denotes tuning deviation from 12-tone equal temperament in 20-cent resolution ($f \in \{1,\ldots,5\}$, with $f = 3$ denoting perfect equal temperament), o denotes the vowel type, and v denotes the voice type (soprano, alto, tenor, baritone, or bass).

The contribution of specific singer subjects from the training dictionary is modelled by $P_t(s|p)$, i.e. the singer contribution per pitch over time. $P_t(f|p)$ is the tuning deviation per pitch over time and finally $P_t(o|p)$ is the time-varying vowel contribution per pitch.[6] Unlike the work of Schramm and Benetos (2017) (which uses $P_t(v|p)$), this model decomposes the probabilities of pitch and voice type as $P(v)P_t(p|v)$. That is, $P(v)$ can be viewed as a mixture weight that denotes the overall contribution of each voice type to the whole input recording, and $P_t(p|v)$ denotes the pitch activation for a specific voice type (SATB) over time.

The factorisation can be achieved by the expectation-maximisation (EM) algorithm (Dempster et al., 1977), where the unknown model parameters $P_t(s|p)$, $P_t(f|p)$, $P_t(o|p)$, $P_t(p|v)$, and $P(v)$ are iteratively estimated. In the *Expectation* step, we compute the posterior as in Equation (3.15), and in the *Maximisation* step, each unknown model parameter is then updated using the posterior from Equation (3.15) as shown in Equations (3.16)–(3.20).

$$P_t(s,p,f,o,v|\omega) = \frac{P(\omega|s,p,f,o,v)P_t(s|p)P_t(f|p)P_t(o|p)P(v)P_t(p|v)}{\sum_{s,p,f,o,v} P(\omega|s,p,f,o,v)P_t(s|p)P_t(f|p)P_t(o|p)P(v)P_t(p|v)} \tag{3.15}$$

$$P_t(s|p) \propto \sum_{f,o,v,\omega} P_t(s,p,f,o,v|\omega)X_{\omega,t} \tag{3.16}$$

$$P_t(f|p) \propto \sum_{s,o,v,\omega} P_t(s,p,f,o,v|\omega)X_{\omega,t} \tag{3.17}$$

$$P_t(o|p) \propto \sum_{s,f,v,\omega} P_t(s,p,f,o,v|\omega)X_{\omega,t} \tag{3.18}$$

$$P_t(p|v) \propto \sum_{s,f,o,\omega} P_t(s,p,f,o,v|\omega)X_{\omega,t} \tag{3.19}$$

$$P(v) \propto \sum_{s,f,o,p,\omega,t} P_t(s,p,f,o,p|\omega)X_{\omega,t} \tag{3.20}$$

The model parameters are randomly initialised, and the EM algorithm iterates over Equations (3.15)–(3.20). In our experiments, we use 30 iterations, as this ensures that

[6] Although $P_t(o|p)$ is not explicitly used in the proposed approach, it is kept to ensure consistency with the RWC audio dataset (Goto, Hashiguchi, Nishimura, & Oka, 2004) structure (see Section 3.3.2.1.2).

the model will converge; in practice, the model converges after about 18 iterations. In order to promote temporal continuity, we apply a median filter to the $P_t(p|v)$ estimate across time, before its normalisation at each EM iteration, using a filter span of 240ms, a duration of approximately half of one beat in *Allegro* tempo.

3.3.2.1.1 Acoustic Model Output The output of the acoustic model is a semitone-scale pitch activity tensor for each voice type and a pitch shifting tensor, given by $P(p,v,t) = P(t)P(v)P_t(p|v)$ and $P(f,p,v,t) = P(t)P(v)P_t(p|v)P_t(f|p)$ respectively. By stacking together slices of $P(f,p,v,t)$ for all values of p, we can create a 20-cent resolution time-pitch representation for each voice type v as in Equation (3.21), where $f' \in \{0,...,439\}$ denotes pitch in 20-cent resolution.

$$P(f',v,t) = P\left(f' \pmod 5) + 1, \left\lfloor \frac{f'}{5} \right\rfloor + 21, v, t\right) \tag{3.21}$$

The voice-specific 20-cent resolution pitch activation output is given by $P(f',v,t)$, and the overall multi-pitch activations without voice assignment are given by $P(f',t) = \sum_v P(f',v,t)$. The 20-cent resolution multi-pitch activations $P(f',t)$ are converted into multi-pitch detections, represented by a matrix $B(f',t)$, through a binarisation process with a fixed threshold L_{th}. Specifically, pitch activations whose values are greater than L_{th} are set to 1 in matrix B, while all others are set to 0.

This binary matrix $B(f',t)$ is then post-processed in order to obtain more accurate pitch activations. In this step, we scan each time frame of the matrix B, replacing the pitch candidates by the position of spectrogram peaks detected from $X_{\omega,t}$ that are validated by the minimum pitch distance rule shown in Equation (3.22), where $B(f',t)$ represents each binarised pitch activation at time frame t.

$$(\Delta_{peaks}(X_t, B(f',t)) < T_1) \vee (\Delta_{peaks}(X_t, B(f',t-1)) < T_2) \tag{3.22}$$

The function Δ_{peaks} returns the minimum pitch distance between the selected list of peak candidates in X_t and each pitch candidate $B(f',t)$ and $B(f',t-1)$, respectively. The use of the previous frame $(t-1)$ helps to ensure temporal continuity when a pitch candidate is eventually removed by the L_{th} threshold.

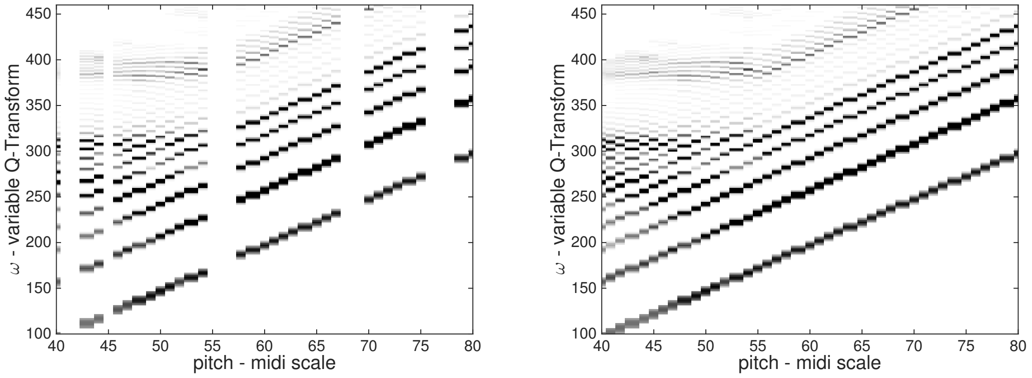
3.3.2.1.2 Dictionary extraction The dictionary $P(\omega|s,p,f,o,v)$ with spectral templates from multiple singers is built based on English pure vowels (monophthongs). The dictionary uses spectral templates extracted from solo singing recordings in the

Musical Instrument Sound subset of the RWC database (RWC-MDB-I-2001 No. 45–49) (Goto et al., 2004). The recordings contain sequences of notes following a chromatic scale, where the range of notes varies accordingly to the tessitura of distinct vocal parts, and each singer sings a scale with each of five distinct English vowels (/a/, /æ/, /i/, /ɒ/, /u/). In total, we have used 15 distinct singers: 9 male and 6 female, consisting of 3 vocalists for each voice type (bass, baritone, tenor, alto, and soprano).

Although the aim of this work is the transcription of vocal quartets, we keep the spectral templates from all five voice types in the dictionary because we do not know in advance the voice types present in each audio recording. This decision allows the dictionary to cover a wider variety of vocal timbres during the spectral decomposition, although not all of the resulting voice assignment probabilities will be used during its integration with the music language model for a single song. Rather, our model dynamically aligns one of the dictionary's voice types to each vocal part in a song. This dynamic dictionary alignment is based on the music language model's voice assignments, and is discussed further in Section 3.3.2.3.

The fundamental frequency (f_0) sequence from each monophonic recording is first estimated using the Probabilistic YIN (PYIN) algorithm (Mauch & Dixon, 2014). Then, a time-frequency representation is extracted using a VQT with 60 bins per octave. A spectral template is extracted for each frame, regarding singer source, vowel type, and voice type. To incorporate multiple estimates from a single pitch, the set of estimates falling in the same pitch bin are replaced by its metrically trimmed mean, discarding 20% of the samples as possible outliers. The use of the metrically trimmed mean aims to reduce the influence of possible pitch inaccuracies obtained from the automatic application of the PYIN algorithm. However, there is no guarantee that the final estimate will be free of eventual outliers. The set of spectral templates are then pre-shifted across log-frequency in order to support tuning deviations for ± 20 and ± 40 cent, and are stored into a 6-dimensional tensor $P(\omega|s,p,f,o,v)$. Due to a lack of data from the chromatic scales, the resulting dictionary $P(\omega|s,p,f,o,v)$ has some pitch templates missing, as shown in Figure 3.10a.

To address this issue, we have investigated different ways to fill out the missing templates in the dictionary, including spectrum estimation by replication (de A. Scatolini, Richard, & Fuentes, 2015; Benetos et al., 2014), linear and nonlinear interpolation, and a generative process based on Gaussian mixture models, inspired by Goto (2004) and Kameoka, Nishimoto, and Sagayama (2007). We have chosen a replication approach, where existing templates belonging to the dictionary are used to fill in



(a)

(b)

Figure 3.10: Example templates from the /a/ vowel utterance of a single singer: (a) original templates from the VQT spectrogram; (b) revised dictionary templates following replication.

the missing parts of the pitch scale, as it has been shown to achieve the best performance (Kirchhoff, Dixon, & Klapuri, 2013). In this approach, the spectral shape of a given pitch p_n is repeated (with the appropriate log-frequency shift) over all subsequent pitches $p \in [p_{n+1}, p_{m-1}]$ until another template is found (the pitch template p_m). Figure 3.10b illustrates the resulting dictionary templates of one singer example (for the vowel /a/) from our dataset, following the above replication process.

3.3.2.2 Music Language Model

The music language model attempts to assign each detected pitch to a single voice based on musicological constraints. It is a variant of the voice separation HMM described in Section 3.2, and only the differences between the original HMM and this variant are described here.

The observed data for the HMM here are notes generated from the acoustic model's binarised 20-cent resolution multi-pitch activations $B(f',t)$, rather than notes read in from a MIDI file. Each activation generates an observed note n with pitch $\text{Pitch}(n) = \lfloor \frac{f'}{5} \rfloor$, onset time $\text{On}(n) = t$, and offset time $\text{Off}(n) = t + 1$ (and therefore a duration $\text{Dur}(n) = 1$). Duplicates are discarded in the case where two 20-cent resolution detections map to the same semitone pitch. O_t represents this set of observed notes at frame t.

3.3.2.2.1 State Space In this version of the HMM, each state again represents a list of monophonic voices. However, here we assume that the number of voices in a song is known a priori. Thus, in the HMM, a state S_t at frame t contains a list of M monophonic voices V_i, $1 \leq i \leq M$, rather than allowing the model to add new voices dynamically. M is set via a parameter, and in this work, $M = 4$. In the initial state S_0, all of the voices are empty.

That this version of the HMM is given the number of voices a priori represents a significant change from the assumptions of the previous version from Section 3.2. In practice, due to noisy pitch detections from the acoustic model (in the form of false positives), such knowledge is necessary to ensure good performance.

3.3.2.2.2 Emission Function A state S_t emits a set of notes with onset at time t, with the constraint that a state containing a voice with a note at onset time t must emit that note. The probability of a state S_t emitting the note set O_t is shown in Equation (3.23), using the voice posterior $P_t(v|p)$ from the acoustic model.

$$P(O_t|S_t) = \prod_{n \in O_t} \begin{cases} P_t(v = i|p = \text{Pitch}(n)) & n \in V_i \in S_t \\ 1 & \text{otherwise} \end{cases} \tag{3.23}$$

Notice that a state is not penalised for emitting notes not assigned to any of its voices. This allows the model to better handle false positives from the multi-pitch detection. For example, if the acoustic model detects more than M pitches, the state is allowed to emit the corresponding notes without penalty. We do, however, penalise a state for not assigning a voice any note during a frame, but this is handled by $\Psi(W)$ from Equation (3.25), described in the following section.

3.3.2.2.3 Transition Function While the previous HMM's valid transitions were enumerated by use of the deterministic emission function, that is not possible here with the new probabilistic emission function. Rather, a state S_{t-1} has a transition to state S_t if and only if each voice $V_i \in S_{t-1}$ can either be transformed into the corresponding $V_i \in S_t$ by assigning to it a single note with onset time t, or if it is identical to the corresponding $V_i \in S_t$.

This transition from S_{t-1} to S_t can be represented by the variable T_{S_{t-1},N_t,W_t}, where S_{t-1} is the original state, N_t is a list of every note with onset time t assigned to a voice in S_t, and W_t is a list of integers, each representing the voice assignment index for the corresponding note N_t. Specifically, N_t and W_t are of equal length, and the ith integer in W_t represents the index of the voice to which the ith note in N_t is assigned in S_t. Notice that here, N_t only contains those observed notes which are assigned to a voice in S_t, rather than all observed notes. Also notice that, because new voices cannot by dynamically added, each element in W_t is non-negative, unlike in the previous HMM.

The HMM transition probability $P(S_t|S_{t-1})$ is defined as $P(T_{S_{t-1},N_t,W_t})$, shown in Equation (3.24). The first term in the equation is a function representing the voice assignment probability, and is defined in Equation (3.25), where the parameter P_v represents the probability that a given voice contains a note in a frame.

$$P(T_{S_{t-1},N_t,W_t}) = P(W_t)\prod_{i=1}^{|N_t|} P(V_{w_i}, n_i) * \text{order}(S_{t-1}, n_i, w_i) \tag{3.24}$$

$$P(W) = \prod_{j=1}^{M} \begin{cases} P_v & j \in W \\ 1 - P_v & j \notin W \end{cases} \tag{3.25}$$

The function $\text{order}(S_{t-1}, n, w)$ is a penalty function used to minimise voice crossings as in the previous HMM; however, two changes have been made. First, because w can no longer be negative, the cases have been simplified. Additionally, it still returns by default 1, but its output is multiplied by a parameter P_{cross}—representing the probability of a voice being out of pitch order with an adjacent voice—rather than $\frac{1}{2}$, for violating each of the following cases:

1. $w > 1$ and $\text{Pitch}(V_{w-1}) > \text{Pitch}(n)$

2. $w < M$ and $\text{Pitch}(V_{w+1}) < \text{Pitch}(n)$

Cases 1 and 2 apply when a note is out of pitch order with the preceding or succeeding voice in the state respectively.

$P(V,n)$ represents the probability of a note n being assigned to a voice V, and is the product of a pitch score and a gap score as in Equation (3.26). Notice that the $w < 0$ case has been removed compared with Equation (3.7).

$$P(V,n) = \text{pitch}(V,n) * \text{gap}(V,n) \tag{3.26}$$

The pitch score, used to minimise melodic jumps within a voice, is computed similarly to the previous HMM, as shown in Equation (3.27), where the Gauss function used is again the Gaussian window function defined in Equation (3.9). The gap score is used to prefer temporal continuity within a voice, and is computed similarly to the

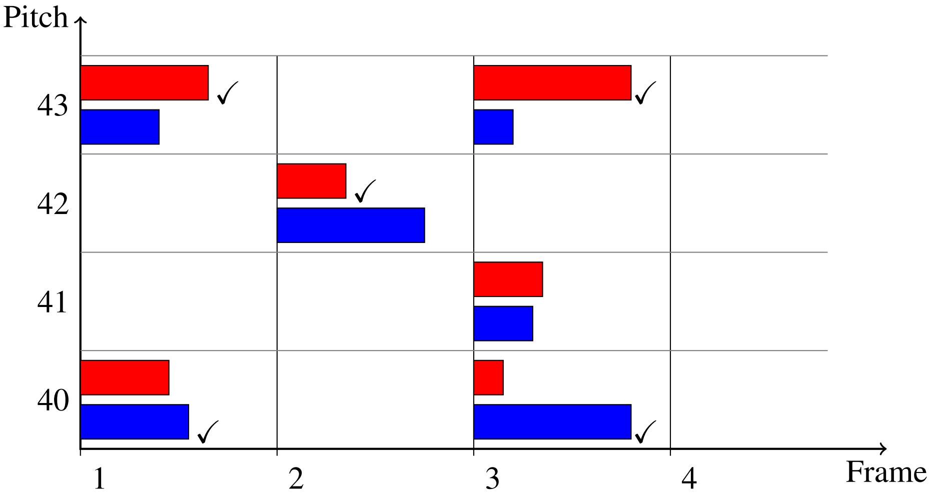


Figure 3.11: An example of an input to the music language model given a simple song with only two voices. Here, for each detected pitch, there are two bars, representing the relative value of $P_t(p|v)$ for each voice (noted by colour) at that frame. The ground truth voice assignment for each detected note is given by a check mark next to the bar representing the correct voice. Notice that there is a false positive pitch detection at pitch 41 at frame 3.

previous HMM using Equation (3.28). Both $\text{pitch}(V,n)$ and $\text{gap}(V,n)$ return 1 if V is empty.

$$\text{pitch}(V,n) = \text{Gauss}(\text{Pitch}(n) - \text{Pitch}(V), \sigma_p) \tag{3.27}$$

$$\text{gap}(V,n) = \max\left(\ln\left(-\frac{\text{On}(n) - \text{Off}(\text{last}(V))}{\sigma_g} + 1\right) + 1, g_{min}\right) \tag{3.28}$$

3.3.2.2.4 Inference To find the most likely final state given our observed note sets, we use the Viterbi algorithm (Viterbi, 1967) with beam search with beam size b. That is, after each iteration, we save only the b most likely states given the observed data to that point, in order to handle the complexity of the HMM. An simple two-voice example of the HMM being run discriminatively can be found in Figures 3.11 and 3.12.

Figure 3.11 shows example input pitch detections, where empty grid cells represent pitches which have not passed the PLCA's post-processing binarisation step, and the bars in the other cells represent relative values of $P_t(p|v)$ for each colour-coded voice. There is a check mark next to the bar representing the ground-truth voice assignment for each detected pitch. Notice that there is no check mark in the cell representing pitch 41 at frame 3, indicating a false positive pitch detection.

Figure 3.12 shows the HMM decoding process of the input from Figure 3.11, using a beam size of 2 and 2 voices. Notes are represented as "[pitch, frame]", and are colour-coded based on their ground truth voice assignment. Again, notice false positive pitch detection $[41,3]$. In this figure, the emission sets O_t are shown on the bottom, and the boxes below each O_t node list the emitted notes in decreasing pitch order. Meanwhile, the voices contained by a state at each timestep are listed in the boxes above each S_t node, where voices are listed in decreasing pitch order, and are separated by braces. The most likely state hypothesis at each timestep is on the bottom row, and each state box (except for S_0) has an incoming arrow indicating which prior state hypothesis was used to transition into that state. Those state hypotheses with an entirely correct voice assignment are represented by a thick border.

Initially, S_0 contains 2 empty voices. Next, O_1 is seen, and the most likely voice assignment is also the correct one, assigning the pitches to the voices in decreasing pitch order. The 2nd hypothesis for S_1 is very unlikely: the two voices are out of pitch order with each other, and its values of $P_t(p|v)$ are lower than the correct assignments. Thus, once O_2 is seen at frame 2, that hypothesis drops out and both hypothesis S_2 states transition from the most likely S_1 state. However, due to noisy $P_t(p|v)$ estimates from the PLCA, the most likely S_2 contains an incorrect assignment for the note $[42,2]$, while the 2nd S_2 hypothesis is correct. In S_3, however, these hypotheses flip back, resulting in the correct overall voice assignment for this example input. Notice that the false positive pitch detection $[41,3]$ is not assigned to any hypothesis state since its values of $P_t(p|v)$ are relatively small. Meanwhile, the $P_t(p|v)$ estimates from the PLCA for the other 2 pitches are quite good, and allow the HMM to correct itself (assuming good parameter settings), judging that the voice $\{[43,1], [42,2], [43,3]\}$ in the higher voice is more likely than the voice $\{[40,1], [42,2], [40,3]\}$ in the lower voice, even given the noisy $P_t(p|v)$ estimates for the note $[42,2]$.

3.3.2.3 Model Integration

In this section, we describe the integration of the acoustic model and the music language model into a single system which jointly performs multi-pitch detection and voice assignment from audio. The pitch activations $P_t(p|v)$ for each voice type from the PLCA dictionary (bass, baritone, tenor, alto and soprano) are quite noisy, resulting in very low accuracy for voice assignment, as can be seen from our results (Table 3.7,

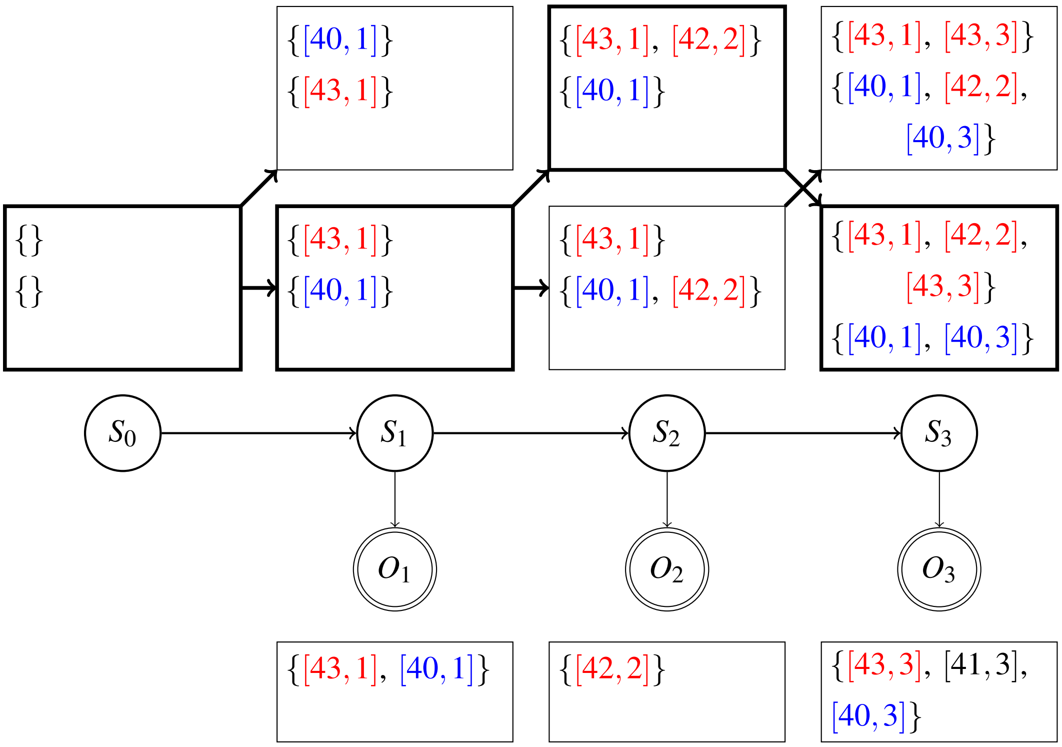


Figure 3.12: An example of the music language model being run on the detected pitches from Figure 3.2 with a beam size of 2 and 2 voices. Notes are represented as "[pitch,frame]", and are colour-coded based on their ground truth voice assignment. The observed note sets are listed beneath each O_t. Notice the false positive pitch detection $[41,3]$ in O_3. The two most likely state hypotheses at each step are listed in the large rectangles above each state S_t, where the voices are notated with braces. The most likely state hypothesis at each step appears on the bottom row, and each state has an incoming arrow indicating which prior state hypothesis was used to transition into that state. Those state hypotheses with an entirely correct voice assignment are represented by a thick border.

row VOCAL4-MP). However, we have found that a good prior distribution for $P_t(p|v)$ can drive the spectrogram factorisation towards a more meaningful voice assignment. This prior is given by the music language model, and its integration into the system pipeline is performed in two stages.

Since multi-pitch detections from the acoustic model are the input for the music language model, spurious detections can result in errors during the voice separation process. Therefore, in the first stage, we run the EM algorithm using only the acoustic model from subSection 3.3.2.1 for 15 iterations to allow for convergence to stable multi-pitch detections. Next, the system runs for 15 more EM iterations, this time also using the music language model. During each EM iteration in this second stage, the acoustic model is run first, and then the language model is run on the resulting multi-pitch detections. To intergrate the two models, we apply a fusion mechanism inspired by the one used by Giannoulis, Benetos, Klapuri, and Plumbley (2014) to improve the acoustic model's pitch activations based on the resulting voice assignments.

The output of the language model is introduced into the acoustic model as a prior to $P_t(p|v)$. During the acoustic model's EM updates, Equation (3.19) is modified as:

$$P_t^{new}(p|v) = \alpha P_t(p|v) + (1-\alpha)\phi_t(p|v), \tag{3.29}$$

where α is a weight parameter controlling the effect of the acoustic and language model and ϕ is a hyperparameter defined as:

$$\phi_t(p|v) \propto P_t^a(p|v)P_t(p|v). \tag{3.30}$$

$P_t^a(p|v)$ is calculated from the most probable final HMM state $S_{t_{max}}$ using the pitch score pitch(V,n) from the HMM transition function of Equation (3.27). For V, we use the voice $V_v \in S_{t_{max}}$ as it was at frame $t-1$, and for n, we use a note at pitch p. The probability values are then normalised over all pitches per voice. The pitch score returns a value of 1 when the V is an empty voice (thus becoming a uniform distribution over all pitches). The hyperparameter of Equation (3.30) acts as a soft mask, reweighting the pitch contribution of each voice based on detected pitches from the previous iteration.

Performance depends on a proper alignment between the voice types present in each song and the voice types present in the PLCA dictionary. Therefore, we dynamically assign one of the five voice types present in the dictionary (see Section 3.3.2.1.2) to each of the voices extracted by the music language model. During the first integrated EM iteration, the acoustic model's voice probabilities $P_t(p|v)$ are set to a uniform distribution upon input to the music language model. Additionally, we cannot be sure which voice types are present in a given song, so we run the language model with $M = 5$. Here, the acoustic model's detections contain many overtones, and we do not want to simply use $M = 4$, because many of the overtones are actually assigned a slightly greater probability than the correct notes by the acoustic model. Rather, the overtones tend to be higher in pitch than the correct notes, and thus are almost exclusively assigned to the fifth voice by the HMM. These decisions combine to allow the music language model to drive the acoustic model towards the correct decomposition without being influenced by the acoustic model's initially noisy voice type probabilities.

After this initial HMM iteration, we make the dynamic dictionary voice type assignments using Equation (3.31).

$$\text{VoiceType}(V_i) = \arg\max_{v} \sum_{p,t} P_t(p|v) P_t^a(p|V_i), \tag{3.31}$$

Each voice V_i from the HMM is assigned the voice type v from the dictionary that gives the greatest correlation between the (initial, non-uniform) PLCA voice probabilities $P_t(p|v)$ and the HMM voice priors $P_t^a(p|V_i)$. This alignment procedure begins with the HMM's lowest voice and performs a greedy search, such that for each subsequent voice, the $\arg\max$ only searches over those dictionary voice types not already assigned to a lower HMM voice. This dynamic dictionary voice type assignment allows the model to decide which voice types are present in a given song at runtime. For all subsequent iterations, this voice type assignment is saved and used during integration. Additionally, the HMM is now run with $M = 4$, and the voice type assignment is used to ensure that the PLCA output $P_t(p|v)$ estimates correspond the correct voice indices in the HMM.

We also place certain constraints on the HMM during its first iteration. Specifically, where O_t is the set notes observed at frame t: (1) if $|O_t| \leq M$, each note in O_t must be assigned to a voice in S_t; and (2) if $|O_t| > M$, the voices in S_t must contain exactly the M most likely pitch activations from O_t, according to $P(p,t)$ from the acoustic model, where ties are broken such that lower pitches are considered more likely (since overtones are the most likely false positives).

The final output of the integrated system is a list of the detected pitches at each time frame which are assigned to a voice in the most probable final HMM state $S_{t_{max}}$, along with the voice assignment for each after the full 30 EM iterations.

3.3.3 Evaluation

3.3.3.1 Datasets

We evaluate the proposed model on two datasets of *a cappella* recordings: one of 26 Bach Chorales[7] and another of 22 Barbershop quartets,[8] in total 104 minutes. Each file is in wave format with a sample rate of 22.05 kHz and 16 bits per sample. Each recording has four distinct vocal parts, with one part per channel. The recordings from the Barbershop dataset each contain four male voices, while the Bach Chorale recordings each contain a mixture of two male and two female voices.

A frame-based pitch ground truth for each vocal part was extracted using a monophonic pitch tracking algorithm (Mauch & Dixon, 2014) on each individual monophonic track with default settings. Experiments are conducted using the mix down of each audio file with polyphonic content, not the individual tracks.

3.3.3.2 Evaluation Metrics

We evaluate the proposed system on both multi-pitch detection and voice assignment using the frame-based precision, recall and F-measure as defined in the MIREX multiple-F0 estimation evaluations (Bay, Ehmann, & Downie, 2009), with a frame hop size of 20 ms.

The F-measure obtained by the multi-pitch detection is denoted as F_{mp}, and for this, we combine the individual voice ground truths into a single ground truth for each recording. For voice assignment, we simply use the individual voice ground truths and define voice-specific F-measures of F_s, F_a, F_t, and F_b for each respective SATB vocal part.[9] We also define an overall voice assignment F-measure F_{va} for a given recording as the arithmetic mean of its four voice-specific F-measures. Statistical significance is calculated using a two-tailed t-test.

We use these F-measures rather than the graph-based F-measure from Section 3.2.3.3, because there can be many false positives from multi-pitch detection, and these are not easy to reconcile with the graph-based metric.

[7] The Bach Chorales were acquired from `http://www.pgmusic.com/bachchorales.htm`.

[8] The Barbershop Quartets were acquired from `http://www.pgmusic.com/barbershopquartet.htm`.

[9] Although the Barbershop recordings may not contain the voices SATB, for evaluation, we consider the lowest voice bass, followed by tenor, alto, and soprano, no matter the technical voice type.

3.3.3.3 Training

To train the acoustic model, we use recordings from the RWC dataset (Goto et al., 2004) to generate the 6-dimensional dictionary of log-spectral templates specified in Section 3.3.2.1, following the procedure described in Section 3.3.2.1.2. For the post-processing refinement step of the acoustic model, we use $T_1 = 1$ and $T_2 = 3$, based on density distributions of $|\Delta_{peaks}|$, estimated from measurements in our datasets using the pitch ground truth.

For the HMM, where applicable, we use the values reported in Table 3.3, trained on the fugues (row Inventions), except that we double the value of σ_p to 8 and use a larger beam size of 50 to better handle noise from the acoustic model. We also introduce two new parameters to the system: the voice crossing probability P_{cross} and the voice assignment probability P_v. We use MIDI files of 50 Bach Chorales[10] (none of which appear in the test set), splitting the notes into 20 ms frames, and measure the proportion of frames in which a voice was out of pitch order with another voice, and the proportion of frames in which each voice contains a note. This results in values of $P_{cross} = 0.006$ and $P_v = 0.99$, which we use for testing.

To train the model integration weight α, we use a grid search on the range $[0.1, 0.9]$ with a step size of 0.1, maximising F_{va} for each dataset. Similarly, the value of the threshold L_{th} that is used for the binarisation of the multi-pitch activations in Section 3.3.2.1.1 is based on a grid search on the range $[0.0, 0.1]$ with a step size of 0.01, again maximising F_{va} for each dataset. To avoid overfitting, we employ cross-validation, using the parameter settings that maximise the Chorales' F_{va} when evaluating the Barbershop Quartets, and vice versa; nonetheless, the resulting parameter settings are the same for both datasets: $\alpha = 0.1$ and $L_{th} = 0.01$.

3.3.3.4 Results

We use five baseline methods for evaluation: Vincent et al. (2010), which uses an adaptive spectral decomposition based on NMF; Pertusa and Iñesta (2012), which selects candidates among spectral peaks, validating candidates through additional audio descriptors; Schramm and Benetos (2017), a PLCA model for multi-pitch detection from multi-singers, similar to the acoustic model of our proposed system, although it also includes a binary classifier to estimate the final pitch detections from the pitch activations; as well as two multi-pitch detection methods from the Essentia library

[10] MIDI files available at `http://kern.ccarh.org/`.

Model	Bach Chorales	Barbershop Quartets
Klapuri (2006)	54.62 (3.00)	48.24 (4.50)
Salamon and Gomez (2012)	49.52 (5.18)	45.22 (6.94)
Vincent et al. (2010)	53.58 (6.27)	51.04 (8.52)
Pertusa and Iñesta (2012)	67.19 (3.82)	63.85 (6.69)
Schramm and Benetos (2017)	71.03 (3.33)	70.84 (6.17)
VOCAL4-MP	63.05 (3.12)	59.09 (5.07)
VOCAL4-VA	**71.76 (3.51)**	**75.70 (6.18)**

Table 3.6: Multi-pitch detection results, where standard deviations are shown in parentheses. The post-processing refinement step described in Section 3.3.2.1.1 was also run on the output of all cited methods. VOCAL4-MP represents our proposed method with the acoustic model only, while VOCAL4-VA refers to our fully integrated model.

(Bogdanov et al., 2013): Klapuri (2006), which sums the amplitudes of harmonic partials to detect pitch presence; and Salamon and Gomez (2012), which uses melodic pitch contour information to model pitch detections. For all five of these methods, we also run the post-processing refinement step described in Section 3.3.2.1.1 on their output. The above systems are evaluated against two versions of our proposed model: VOCAL4-MP, using only the acoustic model described in Section 3.3.2.1; and VOCAL4-VA, using the fully integrated model.

From the multi-pitch detection results in Table 3.6, it can be seen that our integrated model VOCAL4-MP achieves the highest F_{mp} on both datasets, and it outperforms all other models significantly ($p < .05$) for both datasets. The fact that VOCAL4-VA outperforms VOCAL4-MP by so much indicates that the music language model is indeed able to drive the acoustic model to a more meaningful factorisation.

For voice assignment, using each baseline method above which does not output any voice assignment information (Klapuri, 2006; Salamon & Gomez, 2012; Vincent et al., 2010; Pertusa & Iñesta, 2012), we run our music language model once on its output with default settings and $M = 4$, after the post-processing refinement step. Meanwhile, for Schramm and Benetos (2017), as well as VOCAL4-MP, the voice assignments are derived from each model's probabilistic voice assignment estimates ($P_t(v|p)$ for Schramm and Benetos (2017), and $P_t(p|v)$ for VOCAL4-MP).

The voice assignment results are shown in Table 3.7, where it can be seen that VOCAL4-VA outperforms the other models, suggesting that a language model is nec-

Model	Bach Chorales				
	F_{va}	F_s	F_a	F_t	F_b
Klapuri (2006)	28.12 (4.38)	24.23 (10.28)	22.98 (11.85)	29.35 (12.43)	35.92 (10.97)
Salamon and Gomez (2012)	24.83 (5.31)	30.03 (12.63)	25.24 (10.92)	21.09 (9.91)	22.95 (9.30)
Vincent et al. (2010)	18.30 (4.87)	13.43 (7.03)	15.52 (6.50)	17.14 (6.77)	27.10 (8.44)
Pertusa and Iñesta (2012)	44.05 (4.60)	40.18 (11.28)	43.34 (7.38)	41.54 (7.02)	50.56 (6.16)
Schramm and Benetos (2017)	20.31 (3.40)	20.42 (5.36)	21.27 (4.75)	14.49 (1.37)	25.05 (2.12)
VOCAL4-MP	21.84 (9.37)	12.99 (11.23)	10.27 (10.13)	22.72 (6.72)	41.37 (9.41)
VOCAL4-VA	**56.49 (10.48)**	**52.37 (12.92)**	**49.13 (11.22)**	**53.10 (11.71)**	**71.38 (6.06)**

Model	Barbershop Quartets				
	F_{va}	F_s	F_a	F_t	F_b
Klapuri (2006)	20.90 (5.79)	2.53 (4.82)	29.02 (13.25)	7.94 (7.48)	44.09 (14.26)
Salamon and Gomez (2012)	20.38 (6.61)	11.14 (10.27)	35.14 (14.04)	8.44 (8.22)	26.81 (13.69)
Vincent et al. (2010)	19.13 (8.52)	10.20 (8.25)	17.97 (9.03)	15.93 (8.85)	32.41 (12.41)
Pertusa and Iñesta (2012)	37.19 (8.62)	30.68 (13.94)	**36.15 (11.70)**	29.15 (13.90)	52.78 (10.37)
Schramm and Benetos (2017)	23.98 (4.34)	24.45 (6.36)	31.61 (6.79)	13.55 (2.18)	26.34 (2.03)
VOCAL4-MP	18.35 (7.56)	2.40 (5.54)	10.56 (13.92)	16.61 (7.31)	43.85 (3.46)
VOCAL4-VA	**49.06 (14.65)**	**41.78 (18.78)**	34.62 (16.29)	**35.59 (16.93)**	**84.25 (6.58)**

Table 3.7: Voice assignment results, where standard deviations are shown in parentheses. The post-processing refinement step described in Section 3.3.2.1.1 was also run on the output of all cited methods. For those which do not output any voice assignment information Klapuri (2006); Salamon and Gomez (2012); Vincent et al. (2010); Pertusa and Iñesta (2012), the music language model was run once on its output with default settings and $M = 4$. VOCAL4-MP represents our proposed method with the acoustic model only. For VOCAL4-MP and Schramm and Benetos (2017), voice assignments are derived from each model's probabilistic voice assignment estimates ($P_t(v|p)$ for Schramm and Benetos (2017), and $P_t(p|v)$ for VOCAL4-MP). VOCAL4-VA refers to our fully integrated model.

essary for the task. It is also clear that integrating the language model as we have, rather than simply including one as a post-processing step, leads to greatly improved performance. Specifically, notice that the difference in performance between our model and the baseline methods is much greater for voice separation than for multi-pitch detection, even though we applied our language model to those baseline methods' results as post-processing.

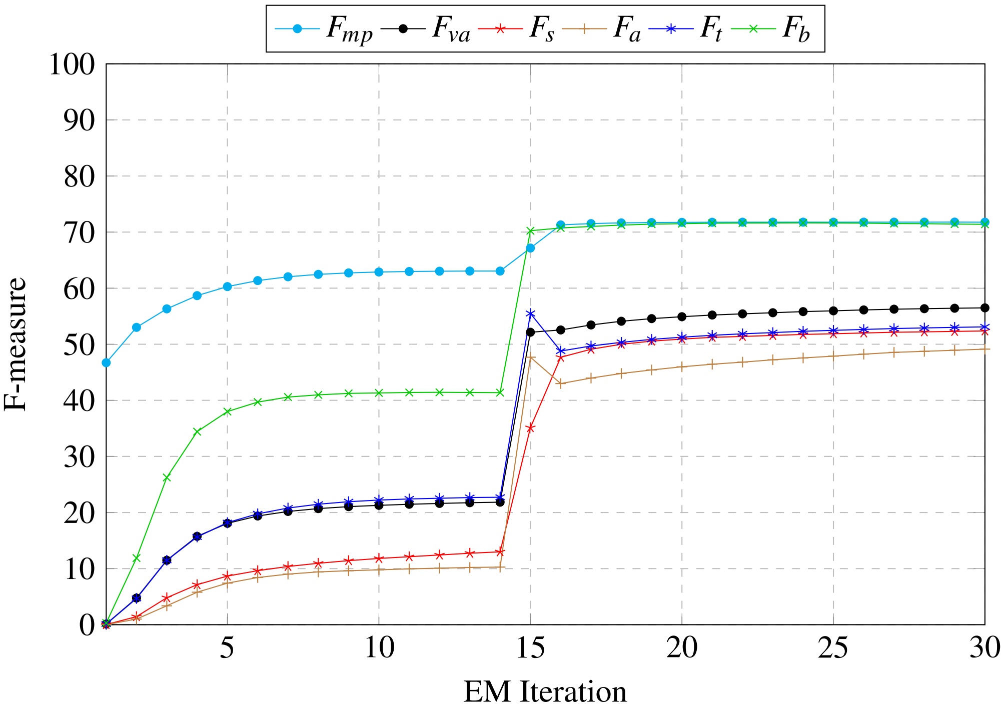
Also interesting to note is that our model performs significantly better on the bass voice than on the other voices ($p < .05$). While this is also true of many of the baseline methods, for none of them is the difference as great as with our model. Overtones are a major source of errors in our model, and the bass voice avoids these since it is almost always the lowest voice.

A further investigation into our model's performance can be found in Figure 3.13, which shows all of the VOCAL4-VA model's F-measures, averaged across all songs in the corresponding dataset after each EM iteration. The first thing to notice is the large jump in performance at iteration 15, when the language model is first integrated into the process. This jump is most significant for voice assignment, but is also clear for multi-pitch detection. The main source of the improvement in multi-pitch detection is that the music language model helps to eliminate many false positive pitch detections using the integrated pitch prior. In fact, the multi-pitch detection performance improves again after the 16th iteration, and then remains relatively stable throughout the remaining iterations.

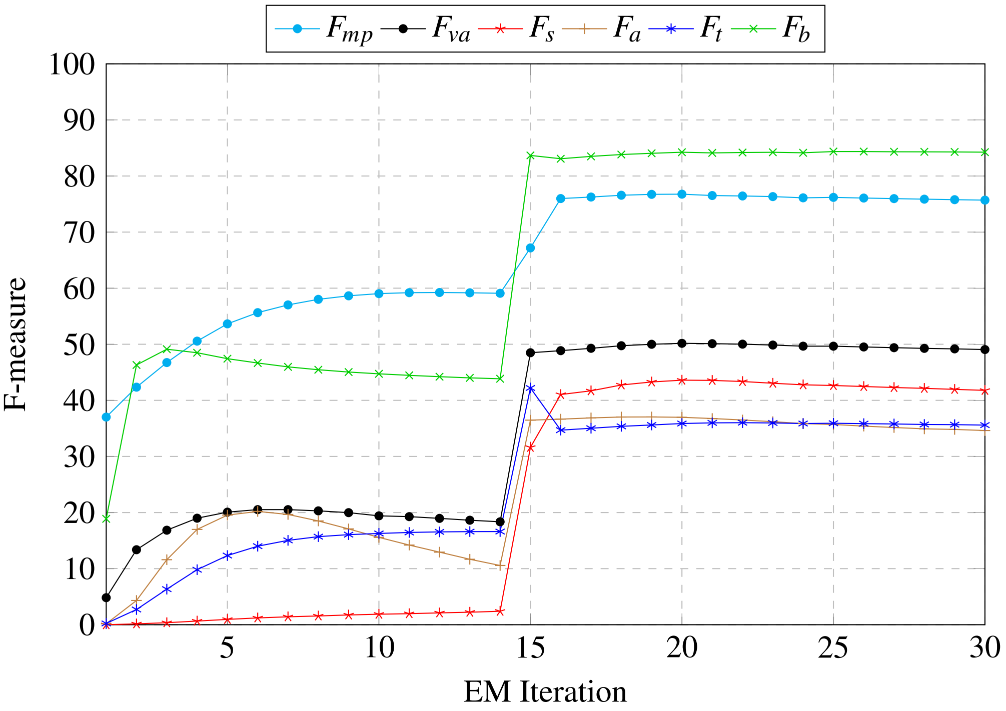
The voice assignment results follow a similar pattern, though without the additional jump in performance after iteration 16. In the Bach Chorales, the voice separation performance even continues to improve until the end of all 30 iterations. For the Barbershop Quartets, however, the performance increases until iteration 20, before decreasing slightly until the end of the process. This slight decrease in performance over the final 10 iterations is due to the alto and soprano voices: F_b and F_t each remain stable over the final 10 iterations, while F_a and F_s each decrease. This difference is likely explained by the acoustic model not being able to properly decompose the alto and soprano voices. The Barbershop Quartets have no true female voices (i.e., each part is sung by a male vocalist), but the template dictionary's contains only three voice types with male vocalists; thus, at least one part, usually either alto or soprano, must be estimated through a rough approximation of a spectral basis combination of female voices. Such a rough approximation could be the cause of our model's difficulty in decomposing the alto and soprano voices in the Barbershop Quartets.

Figure 3.14 illustrates the output of our proposed system, run on excerpts from both the Bach Chorale (a, left) and Barbershop Quartet (b, right) datasets, for the joint multi-pitch detection and voice assignment tasks. Subfigures (a1) and (b1) show the ground truth, using colour to denote vocal part; (a2) and (b2) show the probabilistic pitch detections from the acoustic model after the 30th EM iteration, summed over all voices ($\sum_{v=1}^{5} P_t(p|v)$), where a darker shade of gray indicates a greater probability; (a3) and (b3) present the final output of the integrated system, again using colour to denote vocal part.

As mentioned earlier, the bass voice assignment outperforms all other voice assign-

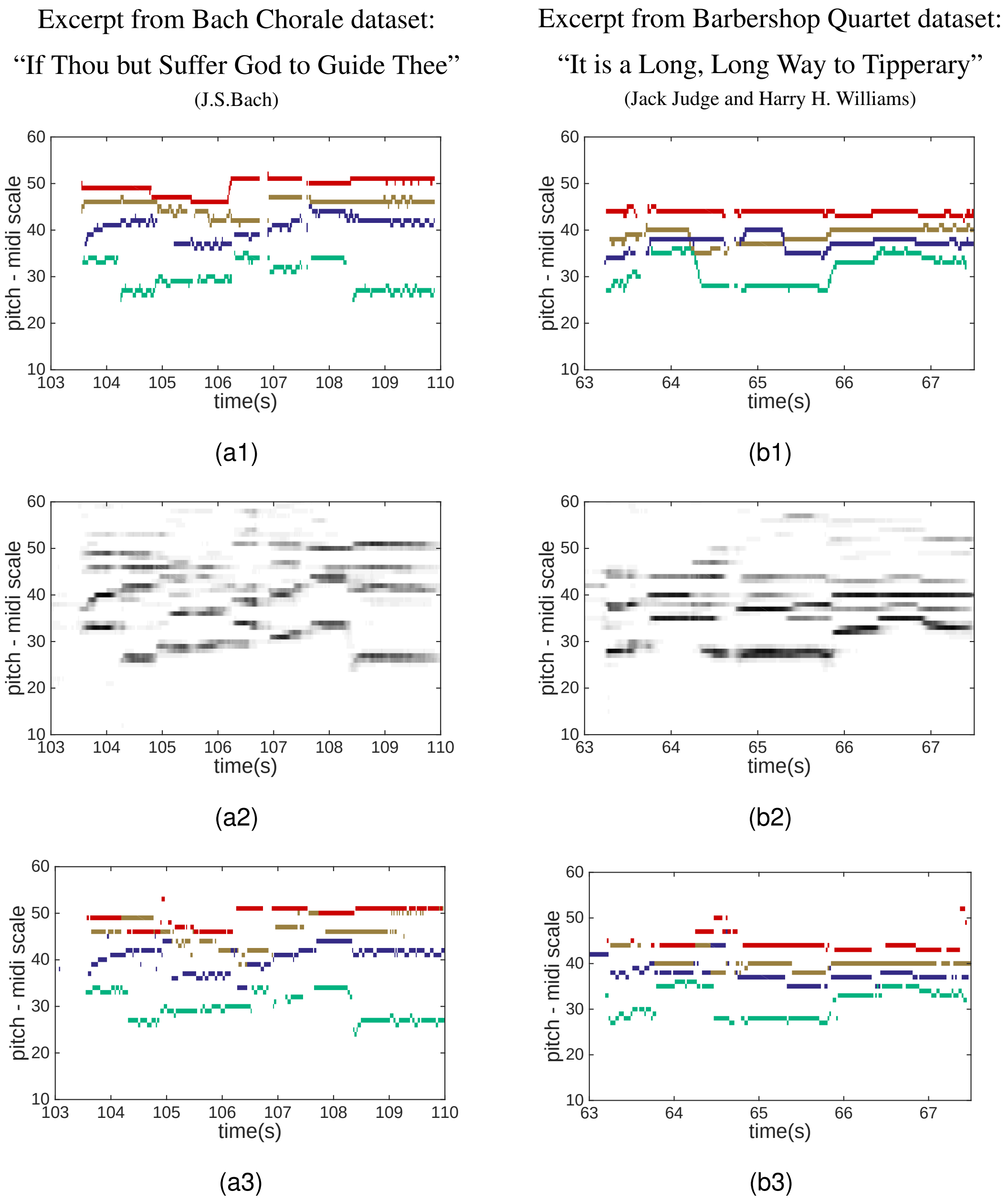


(a) Bach Chorales.



(b) Barbershop Quartets.

Figure 3.13: The VOCAL4-VA model's F-measures after each EM iteration, averaged across all songs in each dataset. The large jump in performance at iteration 15 results from the initial integration of the music language model.



Excerpt from Bach Chorale dataset: “If Thou but Suffer God to Guide Thee” (J.S.Bach)

Excerpt from Barbershop Quartet dataset: “It is a Long, Long Way to Tipperary” (Jack Judge and Harry H. Williams)

(a1) (b1)

(a2) (b2)

(a3) (b3)

Figure 3.14: Example system input and output of excerpts from the Bach Chorale (a, left) and Barbershop Quartet (b, right) datasets. (a1) and (b1) show the ground truth, using colour to denote vocal part; (a2) and (b2) show the probabilistic pitch detections from the acoustic model after the 30th EM iteration, summed over all voices ($\sum_{v=1}^{5} P_t(p|v)$), where a darker shade of gray indicates a greater probability; (a3) and (b3) present the final output of the integrated system, again using colour to denote vocal part.

ments in almost all cases, since false positive pitch detections from the acoustic model often correspond with overtones from lower notes that occur in the same pitch range as the correct notes from higher voices. These overtone errors are most commonly found in the soprano voice, for example at around 105 seconds in the Bach Chorale excerpt and around 64.5 seconds in the Barbershop Quartet excerpt, where (a2) and (b2) clearly show high probabilities for these overtones. It is clear from (a3) and (b3) that such overtone errors in the soprano voice also lead to voice assignment errors in the lower voices since our system can now assign the correct soprano pitch detections to the alto voice, alto to tenor, and tenor to bass.

Another common source of errors (for both multi-pitch detection and voice assignment) is vibrato. The acoustic model can have trouble detecting vibrato, and the music language model prefers voices with constant pitch over voices alternating between two pitches, leading to many off-by-one errors in pitch detection. Such errors are evident throughout the Bach Chorale excerpt, particularly in the tenor voice towards the beginning where our system detects mostly constant pitches (both in the acoustic model output and the final output) while the ground truth contains some vibrato. Also, at the end of both excerpts, there is vibrato present and our system simply detects no pitches rather than the vibrato. This is most evident in the tenor voice of the Bach Chorale, but is also evident in the soprano, alto, and tenor voices of the Barbershop Quartet.

A closer look at errors from both vibrato and overtones can be found in Figure 3.15, which shows pitch detections (red) and ground truth (black) for the soprano voice from an excerpt of “O Sacred Head Sore Wounded” from the Bach Chorales dataset. Here, errors from overtones can be seen around 108.5 seconds, where the detected pitch 54 is the second partial from the tenor voice (not shown), which is at pitch 42 at that time. Errors from vibrato are evident around 107.75 seconds and 108.6 seconds, where the pitch detections remain at a constant pitch while the ground truth switches between adjacent pitches.

3.3.3.4.1 20-cent Resolution To further investigate our model’s performance, especially on vibrato, we present its performance using 20-cent resolution instead of semitone resolution. Specifically, we divide each semitone into five 20-cent wide frequency bins. We convert our integrated model’s final semitone-based output into these bins using a post-processing step: for each detected pitch, we assign it to the 20-cent bin with the maximum $P_t(f|p)$ value from the acoustic model’s final decomposition iteration.

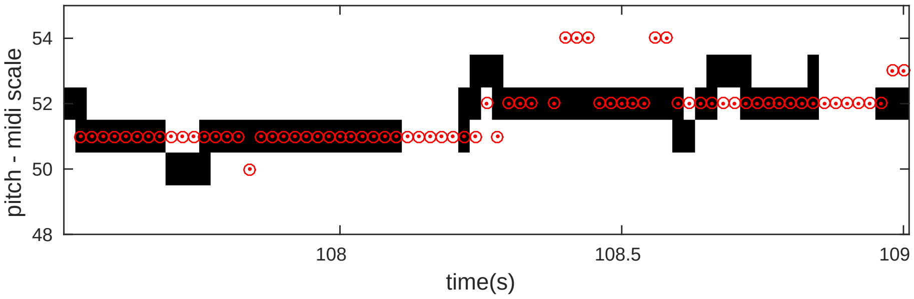


Figure 3.15: Pitch detections (red) and ground truth (black) for the soprano voice from an excerpt of "O Sacred Head Sore Wounded" from the Bach Chorales dataset, showing errors from both vibrato and overtones (from the tenor voice, not shown).

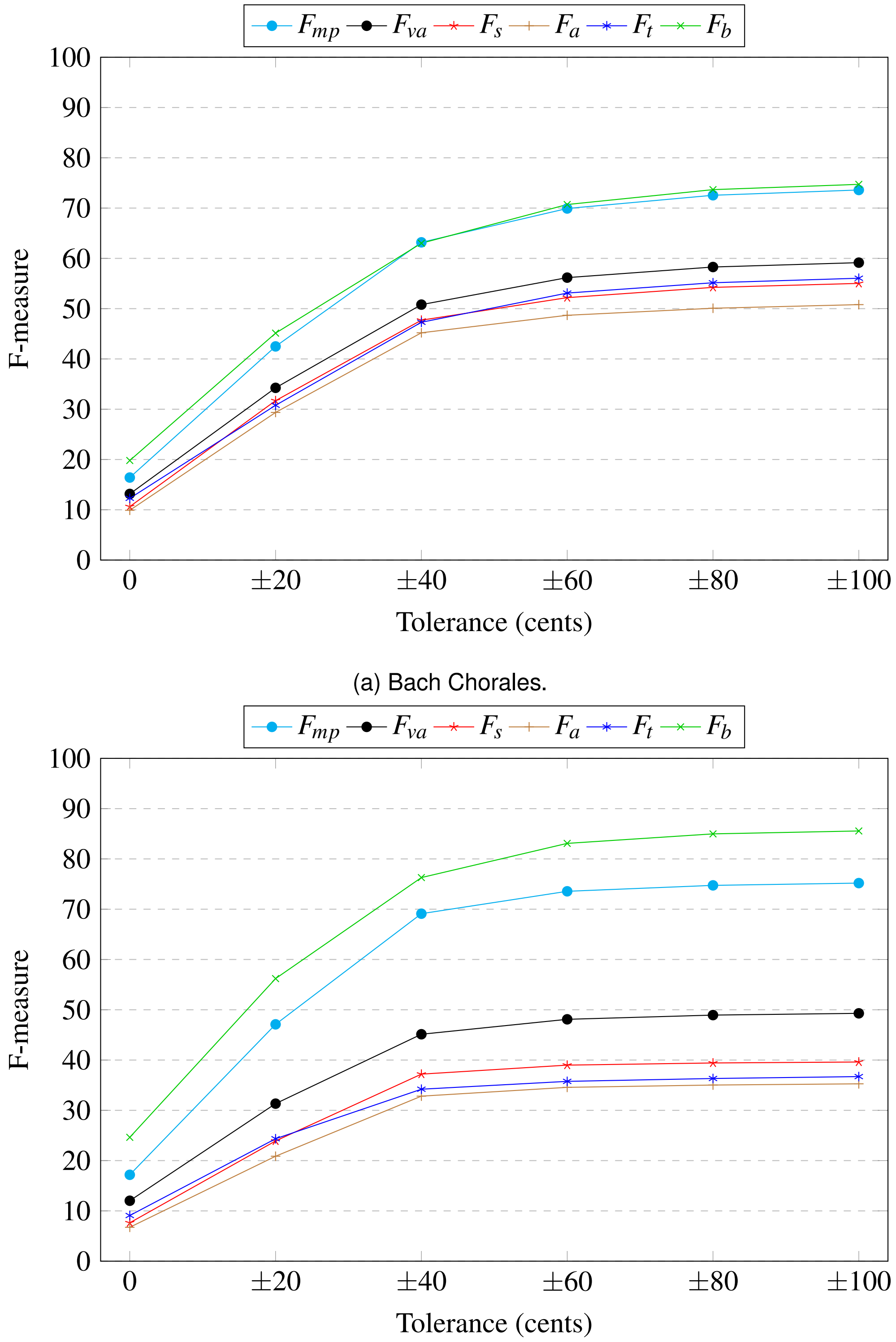
Results are reported in terms of a cent-based pitch tolerance. A tolerance of 0 cents means that a pitch detection will only be evaluated as a true positive if it is in the correct 20-cent bin. A tolerance of ±20 cents means that a pitch detection will be evaluated as a true positive if it is within one bin of the correct bin. In general, a tolerance of ±20*k* cents will count any pitch detection falling within *k* bins of the correct bin as a true positive.

Figure 3.16 illustrates our model's performance using different tolerance levels. In general, our model's semitone-based F-measures lie in between its F-measures when evaluated 20-cent resolution at ±40-cent and ±60-cent tolerance. This does not sound too surprising as a tolerance of ±50 cents would approximate a semitone; however, we would have expected our model's performance with 20-cent resolution to be somewhat better than its performance with semitone resolution, as it should reduce errors associated with vibrato that crosses a semitone boundary. This lack of improvement suggests that our model's difficulty in detecting vibrato is not due simply to semitone crossings, but rather, may be a more fundamental issue of vibrato itself.

3.3.4 Conclusion

In this section, we have presented a system for multi-pitch detection and voice assignment for *a cappella* recordings of multiple singers. It consists of two integrated components: a PLCA-based acoustic model and an HMM-based music language model. To our knowledge, this is the first system to be designed for the task.

We have evaluated our system on both multi-pitch detection and voice assignment on two datasets: one of Bach Chorales, and another of Barbershop Quartets, and we



(a) Bach Chorales.

(b) Barbershop Quartets.

Figure 3.16: Our proposed model's performance on each dataset using pitch tolerance levels from 0 cents up to ± 100 cents.

achieve state-of-the-art performance on both datasets for each task. We have also shown that integrating the music language model improves multi-pitch detection performance compared to a simpler version of our system with only the acoustic model. This suggests, as has been shown in previous work, that incorporating such music language models into other acoustic MIR tasks might also be of some benefit, since they can guide acoustic models using musicological principles.

For voice assignment, while our system performs well given the difficulty of the task, there is certainly room for improvement. As overtones and vibrato constitute the main sources of errors in our system, reducing such errors would lead to a great improvement in the performance of our system. Thus, future work will concentrate on methods to eliminate such errors, for example by post-processing steps which examine more closely the spectral properties of detected pitches for overtone classification and the presence of vibrato. Another possible improvement could be found during the dynamic dictionary voice type assignment step. In particular, running a voice type recognition process as a preprocessing step may result in better performance, and could even eliminate the need for the music language model to be given the number of voices a priori.

We will also investigate the use of incorporating additional information from the acoustic model into the music language model to continue to improve performance. In particular, we currently do not use either the singer subject probabilities $P_t(s|p)$ or the vowel probabilities $P_t(o|p)$ at all, the values of which may contain useful voice separation information. Similarly, incorporating harmonic information such as chord and key information into the music language model could lead to a more informative prior for the acoustic model during integration. Additionally, learning a new dictionary for the acoustic model, for example an instrument dictionary, would allow our system to be applied to different styles of music such as instrumentals or those containing both instruments and vocals, and we intend to investigate the generality of our system in that context.

Another possible avenue for future work is the adaptation of our system to work on the note level rather than the frame level. The music language model was initially designed to do so, but the acoustic model and the integration procedure will have to be adapted as they are currently limited to working on a frame level. Such a note-based system may also eliminate the need for robust vibrato detection, as a pitch with vibrato would then correctly be classified as a single note at a single pitch. An additional benefit to adapting our system to work on the note level would be the ability to incorporate metrical or rhythmic information into the music language model.

3.4 Conclusion

This chapter has presented two new models for voice separation, the division of the notes of a musical performance into streams called voices, and voice assignment, which additionally assigns a label to each voice representing the instrument or part to which those notes belong.

First, in Section 3.2, we have presented an HMM for performing voice separation on MIDI data. We have shown that the proposed model, in addition to being one of only a very few models which can be run on live performance data directly, outperforms baseline methods on multiple corpora of Bach and Haydn compositions, achieving state-of-the-art results on both metronomic and live performance MIDI data as input. It also requires as input only only note onset time, offset time, and pitch; no metrical, harmonic, or other information (such as a maximum number of voices) is required as it is for many other approaches. Additionally, the incrementality of the model allows for it to be used in any number of real-time applications, as it can begin processing before the end of a performance is reached.

In Section 3.3, we have presented a system for multi-pitch detection and voice assignment of *a cappella* recordings, integrating an acoustic model based on PLCA with a music language model in the form of a modified version of the voice separation HMM. Although ours is the first system designed directly for the task, we have compared it against systems comprised of existing multi-pitch detection models with the modified version of the voice separation HMM run as a post-processing step. The experiments show that our system outperforms all other systems, achieving state-of-the-art results on both a corpus of Bach Chorales and another of Barbershop Quartets.

From a computational standpoint, voice separation is commonly used for the preprocessing of data into easier to handle monophonic streams of notes, particularly in the case of MIDI input. In fact, in prior work, voice separation has almost exclusively been addressed in a MIDI setting. To that end, our voice separation model has offered significant improvements over existing approaches in (1) performance, where we achieve state-of-the-art accuracy; (2) the amount of data required as input, where our model eliminates the need for any a priori metrical alignment, harmonic information, or voice count; and (3) its applicability to live performance, where our model has been designed to work directly on live performance, even in real time due to its incrementality.

We have also shown that voice separation has more value than its use simply as a preprocessing step for other tasks. Our joint multi-pitch detection and voice separation system has clearly demonstrated that integrating an acoustic model with our voice separation model significantly improves multi-pitch detection performance compared to using the acoustic model by itself. This proves that a voice separation model can be used as a successful music language model for acoustic MIR tasks.

There is still definite room for improvement on both of our models, however. For example, combining the HMM with a model of rhythmic or metrical structure and running the two jointly could eliminate many of the errors which our model currently makes. This would lead to improved performance on both MIDI voice separation and audio voice assignment (although that would require the additional step of having the acoustic model generate notes instead of frame-based pitch detections).

Chapter 4

Metrical Analysis

This chapter discusses the metrical analysis of music data, which is, in simplest terms, the detection of the underlying metrical structure of a piece of music. Metrical analysis should play a major role in any music language model, allowing the model to interpret a stream of notes as structured, with stressed and unstressed notes, rather than treating each note as equally important. Such an analysis, in particular aligning a performance with a metrical structure, is a necessary component of any complete transcription system, since the time signature, bar lines, and note values all rely directly on this metrical alignment.

Section 4.3 proposes a lexicalised probabilistic context-free grammar (LPCFG) designed for the task of meter detection and alignment from metronomic MIDI data. That the LPCFG is successfully able to detect metrical structures in music again suggests some underlying similarity between language and music, since LPCFGs have been used successfully in various parsing tasks in NLP. In order for the grammar to be run on live performance data, it must be combined with some beat tracking model. Such a beat tracking model is presented in Section 4.4 in the form of an HMM which runs jointly with the LPCFG to perform metrical structure detection and alignment on live performance data, requiring as input only note onset and offset times. This joint HMM and LPCFG follows all of our constraints for a music language model from Chapter 2, being probabilistic, incremental, using no a priori information, and being able to run on live performance data.

This chapter is based on the published works “Meter detection in symbolic music using a lexicalized PCFG” (McLeod & Steedman, 2017) and “Meter detection and alignment of MIDI performance” (McLeod & Steedman, 2018b).

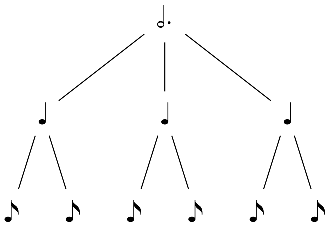


Figure 4.1: The metrical structure of a $\frac{3}{4}$ bar.

4.1 Introduction

The metrical structure of a piece of music can be represented by a tree in which each node represents a single note vale. In common-practice Western music (the subject of our work), the children of each node in the tree divide its duration into some number of equal-length notes (usually two or three) such that every node at a given depth has an equal value. For example, the metrical structure of a single $\frac{3}{4}$ bar, down to the quaver (eighth note) level, is shown in Figure 4.1. Each level of a metrical tree corresponds with an isochronous pulse in the underlying music: bar, beat, and sub beat (from top to bottom). There are theoretically more divisions further down the tree, but as these three levels are enough to unambiguously identify the time signature of a piece, we do not consider any lower. Meter detection refers to the identification of the structure of this tree, while alignment necessitates the additional step of aligning a sequence of trees with the underlying musical performance so that the root of each tree corresponds to a single bar.

Meter detection and alignment are integral components of AMT, particularly when trying to identify the time signature of a given performance, since there is a one-to-one relationship between time signatures and metrical structures. In music, each successive bar may have a different metrical structure than the preceding one; however, such changes in structure are not currently handled by our model, and are left for future work. Our grammar can only be applied to pieces in which each bar is of equal length, has the same number of equal-length beats, and each beat has the same number of equal-length sub beats. That is, it can be applied to any piece where the metrical tree structure under each node at a given level of the tree is identical. In this work, we evaluate our grammar only on the simple and compound meter types $\frac{2}{X}$, $\frac{3}{X}$, $\frac{4}{X}$, $\frac{6}{X}$, $\frac{9}{X}$, and $\frac{12}{X}$ (where X may take any value), and leave more uncommon and irregular meters for future work. Those interested in asymmetric meter detection should refer to Fouloulis, Pikrakis, and Cambouropoulos (2013).

We discuss existing work on meter detection and alignment in Section 4.2. In Section 4.3, we introduce and evaluate our grammar, which is able to perform metrical structure detection and alignment on metronomic data, and in Section 4.4, we incorporate this grammar into an HMM which is able to perform metrical structure detection and alignment on live performance data.

4.2 Existing Work

Most of the early work in the fields of meter detection and alignment involved rule-based, perceptual models. Longuet-Higgins and Steedman (1971) present a model which runs on monophonic metronomic data and uses only note durations, which was later extended by Steedman (1977) to incorporate melodic repetition. Both models were evaluated on full metrical structure detection on the fugues from Bach's Well-Tempered Clavier (WTC). Longuet-Higgins and Lee (1982) describe a somewhat similar model, also to be run on monophonic metronomic data, though only a few qualitative examples are presented in evaluation, and the model is unable to handle syncopation. Spiro (2002) proposes a rule-based, incremental model for metronomic data, combined with a probabilistic n-gram model of bar-length rhythmic patterns, and evaluated on metrical structure detection and alignment on a small corpus of 16 monophonic string compositions by Bach. This remains one of the only successful models for meter detection to use a grammar thus far, though similar grammars have been used for rhythmic and tempo analysis where the meter is given (Takeda, Nishimoto, & Sagayama, 2004, 2007; Nakamura, Yoshii, & Sagayama, 2016). While these rule-based methods show promise, and we base some of our model's principles on them, a more flexible probabilistic model is preferred.

Brown (1993) proposes using auto-correlation for meter detection, in which a promising, though limited, evaluation on meter type and sub beat length detection was shown for 17 metronomic pieces. Meudic (2002) later describes a similar model also using auto-correlation on metronomic MIDI data for the same task. Eck and Casagrande (2005) extend this further, and were the first to use auto-correlation to also perform metrical alignment (though alignment results are limited to synthetic rhythms). They were also the first to do some sort of corpus-based evaluation, though only to classify the meter of a piece as duple or compound. Though auto-correlation has performed well for partial metrical structure detection, there is still a question about whether it can perform metrical alignment, and no work that we have found has yet done so successfully for non-synthetic MIDI data.

Inner metric analysis (IMA) was first proposed for music analysis by Volk (2008), though only as a method to analyse the rhythmic stress of a piece, not to detect the meter of that piece. It requires metronomic MIDI with labelled beats as input, and it involves identifying periodic beats which align with note onsets. Thus, detecting and aligning metrical structure using IMA is a matter of classifying the correct beats as downbeats. It is used by De Haas and Volk (2016), along with some post-processing, to perform meter detection on metronomic MIDI data probabilistically. We were unable to run their model on our data, though they evaluate the model on two datasets, testing both duple or triple classification as well as full metrical structure detection (including phase). However, as the datasets they used are quite homogeneous—95% of the songs in the FMPop corpus are in $\frac{4}{4}$, and 92% of the songs in the RAG corpus (Volk & de Haas, 2013) are in either $\frac{2}{4}$ or $\frac{4}{4}$ time—we have decided not to include a comparison in this work.

Whiteley, Cemgil, and Godsill (2006) perform metrical structure detection and alignment probabilistically from live performance data by jointly modeling tempo, meter, and rhythm; however, the evaluation was very brief, only testing the model on 3 bars of a single Beatles piano performance, and the idea was not used further on MIDI data to our knowledge. Temperley (2007) proposes a Bayesian model for metrical structure detection and alignment of monophonic live performance MIDI data. The general idea is to model the probability of a note onset occurring given the current level of the metrical tree at any time with Bayes' rule. This is combined with a simple Bayesian model of tempo changes, giving a model which can detect and align the full metrical structure of a performance. Temperley (2009) extends this model to work on polyphonic data, combining it into a joint model with a Bayesian voice separator and a Bayesian model of harmony. This joint model performs well on metrical structure detection and alignment on a corpus of piano excerpts, and we compare against it in our work in Section 4.4.

4.3 Tatum-aligned Data

This section proposes a lexicalised probabilistic context-free grammar (LPCFG) designed for meter detection and alignment of metronomic data, an integral component of AMT. The grammar uses rhythmic cues to align a given musical piece with learned metrical stress patterns. Lexicalisation breaks the standard probabilistic context-free grammar (PCFG) assumption of independence of production, and thus, the grammar can model the more complex rhythmic dependencies which are present in musical compositions. Using a novel metric proposed for the task, we show that the grammar outperforms baseline methods when run on metronomic data. The code for the grammar is available at https://github.com/apmcleod/met-detection.

Specifically, our grammar is designed to be run on metronomic MIDI music data, and we present an evaluation where the tatum—the fastest subdivision of the beat (we use demisemiquavers, or 32nd notes, although the grammar does not know which specific level is used)—is given. Thus, the task that our grammar solves is one of detecting and aligning the correct full metrical structure, composed of: (1) meter type (the number of beats per bar and the number of sub beats per beat), (2) phase (the number of tatums which fall before the first full bar), and (3) sub beat length (the number of tatums which lie within a single sub beat).

4.3.1 Proposed Method

For our proposed method, we were careful to make as few assumptions as possible so it can be applied to different styles of music directly (assuming enough training data is available). It is based on a standard PCFG (presented in Section 4.3.1.1) with added lexicalisation as introduced in Section 4.3.1.2. The inference procedure is described in Section 4.3.1.3.

The basic idea of the grammar is to detect patterns of rhythmic stress in a given piece of music with the grammar, and then to measure how well those stress patterns align with learned metrical stress patterns. We use note length to measure rhythmic stress in this work, assuming that long notes will be heard as stressed. This assumption is based on ideas from many of the rule-based methods presented above, and works well. However, there are many other factors of musical stress that our grammar does not capture, such as melody and harmony, which have been found to be helpful for meter detection (Toiviainen & Eerola, 2006), that are left for future work.

4.3.1.1 PCFG

The context-free grammar rules shown in Figure 4.2 is used to construct a rhythmic tree quite similar to the metrical tree from Figure 4.1 above. Each bar of a given piece is first assigned the start symbol S, which can be rewritten as the non-terminal $M_{b,s}$ (representing the meter type), where b is the number of beats per bar and s is the

$$\begin{aligned}
S &\rightarrow M_{b,s} \\
M_{b,s} &\rightarrow B_s \ldots\ B_s \text{ (b times)} \\
B_s &\rightarrow SB \ldots\ SB \text{ (s times)} \mid r \\
SB &\rightarrow r
\end{aligned}$$

Figure 4.2: The grammar rules which form the basis of the PCFG. The subscript b is the number of beats per bar, while s is the number of sub beats per beat. The terminal symbol r can refer to any rhythmic pattern.

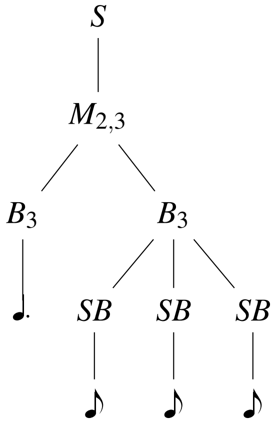


Figure 4.3: An example of the rhythmic tree of a $\frac{6}{8}$ bar with the rhythm ♩. ♪♪♪.

number of sub beats per beat (2 for simple meters and 3 for compound meters). For example, $M_{4,2}$ represents a meter in $\frac{4}{4}$ time, and $M_{2,3}$ represents a meter in $\frac{6}{8}$ time.

A non-terminal $M_{b,s}$ is rewritten as b beat non-terminals B_s. Each beat non-terminal B_s can be rewritten either as s sub beat non-terminals SB or as the terminal r, representing the rhythm of the notes and rests present within that beat. A beat may only be rewritten as r if it contains either (1) no notes or (2) a single note which lasts at least the entire duration of the node (the note may begin before the beat, end after the beat, or both). A sub beat SB must be rewritten as a terminal r, representing the rhythm of the notes and rests present within that sub beat.

An example of the rhythmic tree of a single $\frac{6}{8}$ bar with the rhythm ♩. ♪♪♪ is shown in Figure 4.3. Here, the first beat is been rewritten as a terminal since it is the only note present.

4.3.1.2 Lexicalisation

One downside of using a PCFG to model the rhythmic structure is that PCFGs make a strong independence assumption that is as inappropriate for music as it is for language. Specifically, in a given rhythm, a note can only be heard as stressed or important in contrast with the notes around it, though a standard PCFG cannot model this. A PCFG may see a dotted quarter note and assume that it is a long note, even though it has no way of knowing whether the surrounding notes are shorter or longer, and thus, whether the note should indeed be considered stressed.

To solve this problem, we implement an LPCFG, where each pre-terminal node is assigned a head corresponding to the note beneath it with the longest duration. Strong heads (in this work, those representing longer notes) propagate upwards through the metrical tree to the other non-terminals in a process we call lexicalisation by analogy with lexical head dependency models in NLP. This allows the grammar to model rhythmic dependencies rather than assuming independence as in a standard PCFG, and the pattern of strong and weak beats and sub beats is used to determine the underlying rhythmic stress pattern of a given piece of music.

This head is written $(d;s)$, where d is the duration of the longest note (or, the portion of that note which lies beneath the node), and s is the starting position of that note. When two notes are of equal duration, the one with the earliest starting position is chosen as most important. The character ‘t’ is added to the end of s if that note is tied into (i.e. if the onset of the note lies under some previous node). In the heads, d and s are normalised so that the duration of the node itself is 1. Thus, only heads which are assigned to nodes at the same depth can be compared directly. A node with no notes is assigned the empty head of $(0;0)$.

Once node heads have been assigned, each beat and sub beat non-terminal is assigned a strength of either strong (S), weak (W), or even (E). These are assigned by comparing the heads of siblings in the rhythmic tree. If all siblings’ heads are equal, they are assigned even strength. Otherwise, those siblings with the strongest head are assigned strong strength while all others are assigned weak strength, regardless of their relative head strengths.

Head strength is determined by a ranking system, where heads are first ranked by d such that longer notes are considered stronger. Any ties are broken by s such that an earlier starting position corresponds to greater strength. Any further ties are broken such that notes which are not tied into are considered stronger than those which are.

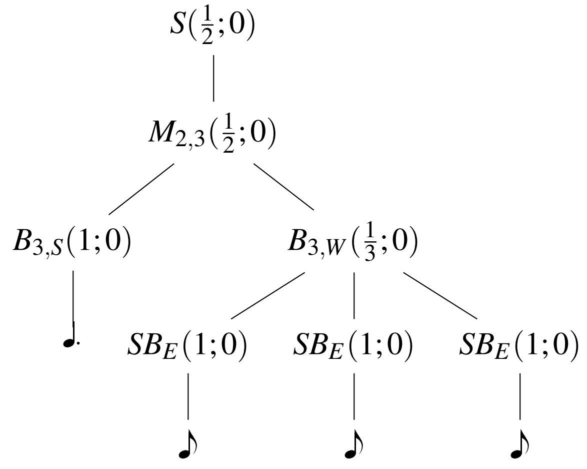


Figure 4.4: An example of the rhythmic tree of a $\frac{6}{8}$ bar with the rhythm ♩. ♪♪♪ including strengths and lexicalisation.

An example of the rhythmic tree of a single $\frac{6}{8}$ bar with the rhythm ♩. ♪♪♪ including strengths and lexicalisation is shown in Figure 4.4. To create this tree, we first take the basic PCFG-generated tree from Figure 4.3 and add heads to each non-terminal node, starting from the bottom. Since the first sub beat contains only one note, and that note's duration is the full sub beat, it is assigned a head of $(1;0)$. The other two sub beats are likewise also assigned a head of $(1;0)$. Moving upwards to the beat level, the first beat is also assigned a head of $(1;0)$. (Remember that each head is normalised so that the duration of the node itself is always 1.) The second beat, however, is assigned a shorter head of $(\frac{1}{3};0)$. The most important note beneath the second beat is the eighth note in its first sub beat (because notes of equal duration are ranked based on their starting position). At the bar level, the head of $(\frac{1}{2};0)$ is chosen based on the dotted quarter note in the first beat, and likewise for the head of $(\frac{1}{2};0)$ at the start symbol. Next, strengths are assigned to each sub beat and beat non-terminal. The heads of each of the sub beats in the second beat are equal, and thus they are all assigned even strength (E). At the beat level, the first beat has a stronger head, and thus the first beat is assigned strong strength (S) while the second beat is assigned weak strength (W).

Certainly the LPCFG does have an advantage over the PCFG in that it has the ability to draw from the wider context of the entire bar when aligning a piece with a metrical structure. However, it is not clear precisely in which cases this context adds value. For example, one might imagine an even more naive model which simply tries to align longer notes with nodes higher up in the metrical tree. This idea is, in fact, very similar to the the main principle behind IMA, mentioned earlier (De Haas & Volk,

	$SB_S\ SB_W$	$SB_E\ SB_E$	$SB_W\ SB_S$
B_S	20.77%	74.08%	5.15%
B_E	10.68%	87.92%	1.40%
B_W	17.61%	44.62%	37.77%

Table 4.1: The LPCFG's learned probabilities for every possible beat to sub beat transition in the context of a $\frac{4}{4}$ bar. The first column lists each possible beat non-terminal while the remaining columns show the probability of every possible sub beat combination.

2016), and in general, it is a good one. However, there are indeed cases in which the wider context offers insight.

To highlight one specific example, we investigate the beat to sub beat transition in the context of a $\frac{4}{4}$ bar. All possible such transitions are listed in Table 4.1, along with their probabilities, and from the differing probability distributions in each row, it quickly becomes clear why the LPCFG's additional context is helpful. The first column lists each possible beat non-terminal while the remaining columns show the probability of every possible sub beat combination. Notably, a weak beat (bottom row) is significantly more likely than either an even or a strong beat to transition into a weak sub beat followed by a strong sub beat (rightmost column). Furthermore, and even a bit counter-intuitively, a weak beat is more likely to transition into a weak sub beat followed by a strong sub beat than it is to transition into a strong sub beat followed by a weak sub beat. Therefore, the LPCFG has learned that, in the context of a weak beat, a long note is actually more likely to occur at the sub beat level than higher up in the metrical tree. Only by using its wider context, and seeing that even longer notes exist elsewhere in the bar, is the LPCFG able to learn such a fact, and it is precisely in cases like this that its context adds value.

4.3.1.3 Performing Inference

Each of the LPCFG rule probabilities are computed as suggested by Jurafsky and Martin (2000), plus additionally conditioning each on the meter type. For example, the replacement $\{M_{2,3}(\frac{1}{2};0) \rightarrow B_{3,S}(1;0)\ B_{3,W}(\frac{1}{3};0)\}$ is modelled by the product of Equations (4.1), (4.2), and (4.3). Equation (4.1) models the probability of a transition given the left-hand-side node's head, while Equations (4.2) and (4.3) model the probability of each child's head given its type and the parent's head.

$$P(M_{2,3} \rightarrow B_{3,S}\, B_{3,W} \mid M_{2,3}, (1/2;0)) \tag{4.1}$$

$$P((1;0) \mid M_{2,3}, B_{3,S}, (1/2;0)) \tag{4.2}$$

$$P((1/3;0) \mid M_{2,3}, B_{3,W}, (1/2;0)) \tag{4.3}$$

The meter type conditioning ensures that the model not prefer one meter type over another based on uneven training data. Specifically, each initial transition $S \rightarrow M_{b,s}$ is assigned a probability of 1. The actual probability values are computed given a training corpus using maximum likelihood estimation with Good-Turing smoothing as described by Good (1953). If a given replacement's head, as modelled by Equations (4.2) and (4.3), is not seen in the training data, we use a backing-off technique as follows. We multiply the probability from the Good-Turing smoothing by a new probability equation, where the meter type is removed (again with Good-Turing smoothing). This allows the grammar to model, for example, the beat-level transitions of a $\frac{9}{8}$ bar using the beat-level transitions of a $\frac{3}{4}$ bar. Note that this does not allow any cross-level calculations where, for example, the beat level of a $\frac{9}{8}$ bar could be modelled by the sub beat level of a $\frac{6}{8}$ bar, though this could be a possible avenue for future work.

The grammar was designed to be used on monophonic melodies, so we use the voices as annotated in the data. Afterwards, only rhythmic information is needed. That is, the grammar uses onset and offset times for each note, and no pitch or velocity information.

The first step in the inference process is to create multiple hypothesis states, each with probability 1, and each corresponding to a different (meter type, sub beat length, phase) triplet, which are treated as latent variables. Meter type corresponds to the specific $M_{b,s}$ which will be used throughout the piece for that hypothesis (there is currently a constraint that pieces do not change time signature during a piece). Sub beat length corresponds to the length of a sub beat of that hypothesis state. This differentiates $\frac{2}{4}$ time from $\frac{2}{2}$ time, for example. Phase refers to how long of an anacrusis a hypothesis will model. That is, how many tatums lie before the first downbeat.

Each state's rhythmic trees are built deterministically, one per voice per bar while that voice is active, throwing out any anacrusis bars. A state's final probability is the product of the probabilities of each of the trees of each of its voices. Specifically, repeating metrical trees are aligned with each ground truth voice with the constraint that these trees match perfectly in meter type, sub beat length, and phase across different voices of the same piece. Each tree is then assigned a probability based on the learned grammar's probabilities, and the product of all of these probabilities is taken as the final probability of a particular (meter type, sub beat length, phase) triplet. After parsing a full piece, the states are ordered by probability and the metrical structure corresponding to the most likely state's (meter type, sub beat length, phase) triplet is picked as the model's resulting metrical alignment.

4.3.1.3.1 Rule of Congruence One final optimisation is made, related to the "rule of congruence" as noted by Longuet-Higgins and Steedman (1971), and further described perceptually by C. Lee (1991). That is, with few exceptions, a composer (at least of classical music), will not syncopate the rhythm before a meter has been established. This means that if the meter has not yet been established, and the underlying rhythm does not match with the metrical structure of a hypothesis state based on its (meter type, sub beat length, phase) triplet, we should be able to remove it. In practice, we allow up to 5 mismatches before eliminating a metrical hypothesis state. In tests, setting this value to anything from 2 to 20 makes no difference in performance; however, the lower the value, the faster the program becomes (and the less room for error there is in the case of a particularly adventurous composer).

A metrical structure hypothesis begins as unmatched, and is considered to be fully matched if and only if both its beat and sub beat levels have been matched. The following paragraphs detail the procedure for the matching of a metrical structure hypothesis given each of the four possible states: fully matched, sub beat matched, beat matched, or unmatched.

If a hypothesis is unmatched, a note which is shorter than a sub beat and does not divide a sub beat evenly is counted as a mismatch. A note which is exactly a sub beat in length is either counted as a mismatch (if it is not in phase with the sub beat), or the hypothesis is moved into the sub beat matched state (otherwise). A note which is between a sub beat and a beat in length is counted as a mismatch. A note which is exactly a beat in length is either counted as a mismatch (if it is not in phase with the beat), or the hypothesis is moved into the beat matched state (otherwise). A note which is longer than a beat, is not some whole multiple of a beat in length, and does not divide a bar evenly is counted as a mismatch.

If a hypothesis is sub beat matched, it now interprets each incoming note based on that sub beat length. That is, any note which is longer than a single sub beat is divided into up to three separate notes (for meter matching purposes only): (1) The part of the note which lies before the first sub beat boundary which it overlaps (if the note begins exactly on a sub beat, no division occurs); (2) The part of the note which lies after the final sub beat boundary which it overlaps (if the note ends exactly on a sub beat, no division occurs); and (3) the rest of the note. After this processing, a note which is longer than a sub beat and shorter than a beat is counted as a mismatch. (Due to note division, this only occurs if the note is two sub beats in length and each beat has three sub beats.) A note which is exactly a beat in length moves the hypothesis into the fully matched state. A note which is longer than a beat and is not some whole multiple of beats is counted as a mismatch.

If a hypothesis is beat matched, it now interprets each incoming note based on that beat length exactly as is described for sub beat length in the previous paragraph. After this processing, a note which is shorter than a sub beat and does not divide a sub beat evenly is counted as a mismatch. A note which is exactly a sub beat in length is either counted as a mismatch (if it is not in phase with the sub beat), or the hypothesis is moved into the fully matched state (otherwise). A note which is longer than a sub beat and shorter than a beat, and which does not align with the beginning or end of a beat, is counted as a mismatch.

Once a metrical hypothesis is fully matched, incoming notes are no longer checked for matching, and the hypothesis will never be removed.

4.3.2 Evaluation

4.3.2.1 Metric

To evaluate our method, instead of just checking whether the top hypothesis' metrical structure is fully correct and properly aligned or not, we want some measure of partial correctness. For instance, if the correct time signature is $\frac{4}{4}$, a guess of $\frac{2}{4}$ should achieve a higher score than a guess of $\frac{6}{8}$. With that in mind, we propose the following metric.

For each of the three levels of the guessed metrical structure, an exact match with a level of the correct metrical structure is counted as a true positive, while a clash—when all of the nodes in a level of the guessed structure cannot be made by some integer multiple or division of nodes from each of the levels of the correct structure—is counted as a false positive. After all three levels have been tested, each of the correct metrical structure's levels which were not matched count as a false negative. Precision, recall, and F-measure can all be computed based on the resulting true positive, false positive, and false negative totals.

Examples of this metric are illustrated in Figure 4.5. Given a correct time signature

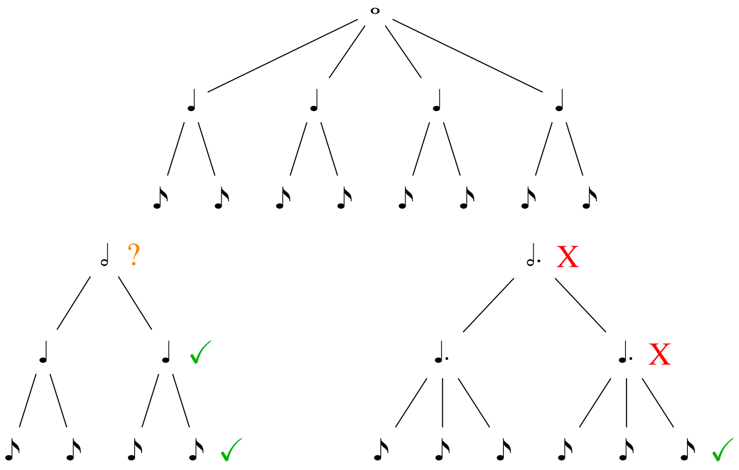


Figure 4.5: Top: The metrical structure of a $\frac{4}{4}$ bar. If $\frac{4}{4}$ is the correct time signature, a guess of $\frac{2}{4}$ with the correct phase (bottom-left) would give $P = 1.0$, $R = 0.67$, and F-*measure* $= 0.8$. A guess of $\frac{6}{8}$ with the correct phase (bottom-right) would give $P = 0.33$, $R = 0.33$, and F-*measure* $= 0.33$.

of $\frac{4}{4}$, and assuming that the phase of the guessed metrical structure is correct,[1] if the guessed time signature is $\frac{2}{4}$, there are only 2 true positives; however, the bar level grouping for $\frac{2}{4}$ does not clash with the metrical structure of $\frac{4}{4}$, so it is not counted as a false positive. There is, however, 1 false negative from the bar level of the $\frac{4}{4}$, giving values of $P = 1.0$, $R = 0.67$, and F-*measure* $= 0.8$. If $\frac{6}{8}$ is guessed instead, the sub beat level again matches, giving 1 true positive. However, both the beat level and the bar level clash (since 1.5 beats of a $\frac{4}{4}$ bar make a single $\frac{6}{8}$ beat, and $\frac{3}{4}$ of a $\frac{4}{4}$ bar gives a $\frac{6}{8}$ bar), giving 2 false positives and 2 false negatives. This gives values of $P = 0.33$, $R = 0.33$, and F-*measure* $= 0.33$. Much lower, and rightfully so.

For evaluation on a full corpus, true positives, false positives, and false negatives are summed throughout the entire corpus to get a global precision, recall, and F-measure. Statistical significance is calculating using a two-tailed t-test.

4.3.2.2 Data

We report our results on two corpora: (1) the 15 Bach Inventions, consisting of 1126 monophonic bars (in which a single bar with two voices counts as two bars), and (2) the much larger set of 48 fugues from the Well-Tempered Clavier (WTC), containing 8835 monophonic bars. These two corpora contain metronomic MIDI files, hand-

[1] An incorrect phase will result in a significantly lower score in all cases presented here.

Method	Inventions	Fugues
$\frac{4}{4}$	0.58	0.45
PCFG	0.61	0.63
LPCFG	**0.63**	**0.80**

Table 4.2: The F-measure of each method for each corpus.

aligned with a demisemiquaver (32nd note) tatum. The notes in each file are split into voices as marked in the corresponding scores.

We use leave-one-out cross-validation within each corpus for learning the probabilities of the grammar. That is, for testing each song in a corpus, we train our grammar on all of the other songs within that corpus. We also tried using cross-validation across corpora by training on the inventions when testing the fugues and vice versa; however, that led to similar but slightly worse results, as the complexities of the rhythms in the corpora are not quite similar enough to allow for such training to be successful. Specifically, there is much more syncopation in the fugues than in the inventions, and thus our grammar would tend to prefer to incorrectly choose meters for the inventions which would result in some syncopation.

4.3.2.3 Results

Our LPCFG is evaluated against two baselines. First, a naive one, guessing $\frac{4}{4}$ time with an anacrusis such that the first full bar begins at the onset of the first note (the most common time signature in each corpus). Second, the PCFG without lexicalisation (as proposed in Section 4.3.1.1, with Good-Turing smoothing and rule of congruence matching). Results are found in Table 4.2, where it can be seen that the LPCFG outperforms all baselines, significantly on the larger fugue corpus ($p < 0.05$). The differences on the smaller inventions corpus are not statistically significant.

It is surprising that the LPCFG does not perform better on the inventions, which are simpler compositions than the fugues. The reason for this lack of improvement seems to be a simple lack of training data, as can be seen in Table 4.3, which shows that as the number of training pieces for each meter type increases, the performance of the LPCFG improves dramatically, showing that more training data in the style of the inventions should continue to improve its performance on that corpus.

Figure 4.6 shows the percentage of pieces in each corpus for which each method achieves 3, 2, 1, or 0 true positives (TPs), and further details exactly what is happening

Meter	Inventions		Fugues	
Type	#	LPCFG	#	LPCFG
$\frac{6}{X}$	0	—	4	0.58
$\frac{3}{X}$	5	0.60	7	0.57
$\frac{2}{X}$	0	—	9	0.89
$\frac{4}{X}$	8	0.71	26	**0.90**
All	15	0.63	48	0.80

Table 4.3: The F-measure of the LPCFG split by meter type. Here, # represents the number of pieces in each meter type, and meter types which occur only once in a corpus are omitted.

on the inventions. The true positive counts correspond with those in our metric, and represent the number of levels of the metrical tree (bar, sub beat, beat) which were matched in both length and phase for each piece. Thus, more true positives corresponds with a more accurate guess.

The improvement on the fugues is clear for the LPCFG. On the inventions, however, the naive $\frac{4}{4}$ model gets 40% of the metrical structures of the inventions exactly correct (3 TPs), while the LPCFG gets only 26.67%. However, the LPCFG gets many more of its guesses mostly or exactly correct (with 2 or 3 TPs), and eliminates fully incorrect guesses (0 TPs) completely. This shows that, even though it may not have had enough data yet to detect and align full metrical structures correctly, it does seem to be learning some sort of structural patterns from what limited data it has.

A specific case where increased training data would benefit the LPCFG is in the 15th fugue from WTC book I, the first bar of which is shown in Figure 4.7. This rhythmic pattern is found throughout the piece, and is a strong indication of a $\frac{6}{8}$ bar, consisting of two even beats, each split into a sub beat pattern of strong, weak, weak. However, our grammar guesses that this piece is in $\frac{4}{4}$ time simply because it has not seen the transition $\{B_{3,E} \rightarrow SB_S\ SB_W\ SB_W\}$ in a $\frac{6}{X}$ meter type in training. This is indicative of the errors we see throughout the data, showing again that with more training data the results will only improve.

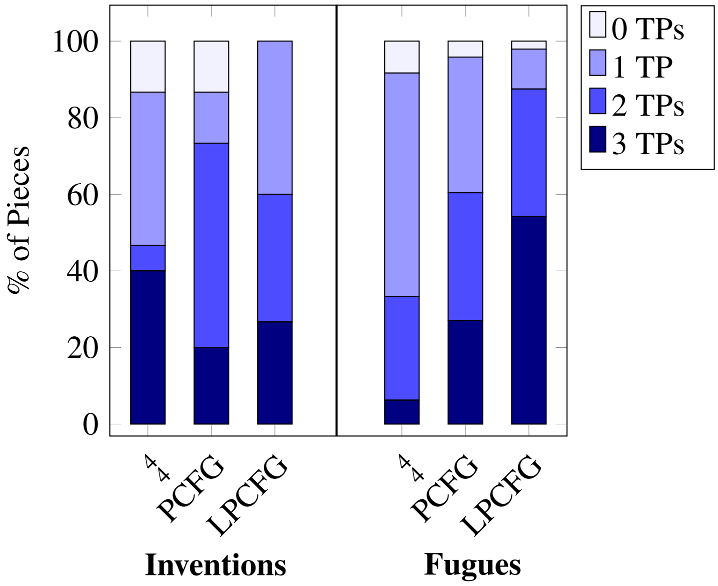


Figure 4.6: The percentage of pieces in each corpus for which each method's metrical structure guess resulted in the given number of true positives.

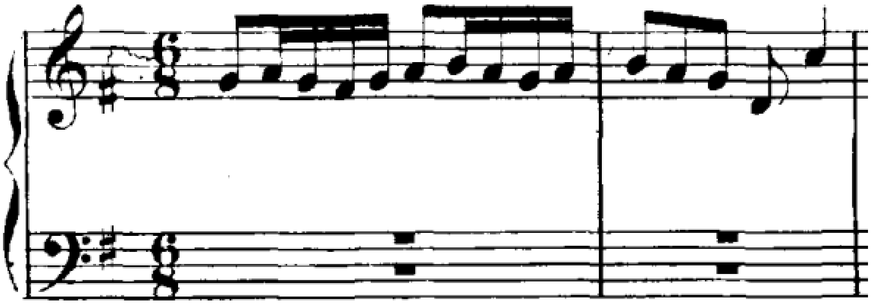


Figure 4.7: The first bar of 15th fugue from WTC book I by Bach (BWV 860).

4.4 With Beat Tracking

In this section, we present an HMM for the metrical alignment of live performance MIDI data. The model needs as input only a list of the notes present in a performance, and detects the underlying metrical structure of the piece, labelling bars, beats, and sub beats in time. We also present an incremental algorithm which can perform inference on the model efficiently. Using a new metric proposed for the task, we show that our model achieves state-of-the-art results on a corpus of metronomic MIDI data, as well as a second corpus of live performance MIDI data. The code for the model described in this section is available at https://www.github.com/apmcleod/met-align.

4.4.1 Proposed Model

The proposed beat tracking model tracks pulses at the tatum level (the fastest subdivision of the beat) of a musical performance based on two musicological principles: (1) the rate of these tatums should be relatively constant without large discontinuities; and (2) notes should lie relatively close to these tatums.

Our model is an HMM where the observed data is the notes of a given piece, grouped into sets. Section 4.4.1.1 describes our model's state space, while Sections 4.4.1.2 and 4.4.1.3 detail its emission and transition functions respectively. Section 4.4.1.4 describes implementation details and additional optimisation procedures used to reduce the search space of the model, as well as to allow it to be more robust in regards to the idiosyncrasies of live performance. An example of our model being run on a sample input is shown and explained in Section 4.4.1.5.

4.4.1.1 State Space

Each state S in our model represents a single bar, containing (1) a list of the tatums from that bar and (2) a metrical hierarchy, describing which of those tatums are beats and sub beats. The list of tatums is represented by $S.t$, where $S.t_i$ is the ith tatum in the bar, and $S.t_{|S.t|}$ is the downbeat of the following bar. The tatums are in increasing time order, where $\text{Time}(S.t_i)$ represents the time of tatum $S.t_i$. A state's metrical hierarchy has some number of tatums per sub beat, sub beats per beat, and beats per bar, as well as an anacrusis length, measured by the number of tatums which fall before the first downbeat of a given piece. In our model, we restrict the number of tatums per sub beat to be 4, although in theory, any number could be used. We also restrict the anacrusis length to be some integer multiple of the number of tatums per sub beat, a simplifying assumption that ensures the first note of each piece will fall on a sub beat. The set of possible sub beat per beat and beat per bar pairs (i.e., time signatures) are taken from the LPCFG described in Section 4.3. A state's tempo, $\mathrm{Tempo}(S)$, is defined as the average length of a beat from its most recent bar.

Each possible initial state S_0 contains no tatums ($|S_0.t| = 0$), and every possible metrical hierarchy is considered equally probable. To reduce our model's search space, we place a restriction on the range of allowed values for $\mathrm{Tempo}(S_1)$: $t_{min} \leq \mathrm{Tempo}(S_1) \leq t_{max}$. Nonetheless, because the possible tatum times for each state are unbounded, our model contains an infinite number of possible states. Thus, instead of predefined emission and transition probabilities, we define emission and transition functions, presented in the following sections.

4.4.1.2 Emission Function

After the initial state (which emits nothing), each state S_i emits a set of notes N_i, containing only notes whose onset times lie between that state's first (inclusive) and last (exclusive) tatum. This set is allowed to be empty. Each emitted note has an onset time $\mathrm{On}(n)$, an offset time $\mathrm{Off}(n)$, a pitch $\mathrm{Pitch}(n)$, and a voice $\mathrm{Voice}(n)$.

The probability of a state S_i to emit the note set N_i is presented as $P(N_i|S_i)$ in Equation (4.4). The first term, calculated entirely by the LPCFG parsing model presented in Section 4.3, is used to prefer generating rhythms which have a high probability according to the grammar, while the second term is used to prefer states whose tatums align closely with the emitted notes. Each emitted note is internally aligned to the nearest tatum by the LPCFG in order to calculate $P(rhythm)$, but this alignment is neither saved nor emitted. Remember that the LPCFG is designed to work directly on monophonic melodies only. Therefore, for polyphonic input, this $P(rhythm)$ term is in fact a product of one probability per voice, each of which is calculated by the LPCFG. For voice assignments, we run the model from Section 3.2 as a preprocessing step.

$$P(N_i|S_i) = P(rhythm) \prod_{n \in N} P(n|S_i.t) \tag{4.4}$$

The $P(n|S_i.t)$ term is calculated as in Equation (4.5), where $\mathcal{N}_1(\mu, \sigma, x)$ conceptually represents a normal distribution with mean μ and standard deviation σ evaluated at x.[2] Thus, $P(n|S_i.t)$ is used to assign higher probabilities to those states which emit

[2] Normal distributions are used in multiple places throughout this model with potentially widely vary-

notes which are closely-aligned with their tatums. In this equation, $closest(S_i.t)$ represents the tatum from S_i whose time is closest to the onset time of the note n.

$$P(n|S_i.t) = \mathcal{N}_1\big(0, \sigma_n, \mathrm{On}(n) - \mathrm{Time}(closest(S_i.t))\big) \tag{4.5}$$

It is important to note that due to the way in which these note sets are grouped by bar, the individual note sets for different paths through the HMM state space for a given piece will not be identical, although the union of all note sets on any given path will equal exactly the set of notes present in the piece. To handle this complication, we introduce a *supervisor* during the HMM decoding process which takes each note individually in onset order, grouping them into note sets and passing the sets to the appropriate hypothesis state at each step. This supervisor is discussed further in Section 4.4.1.4.

4.4.1.3 Transition Function

A state S_{i-1} may transition to a state S_i if and only if: (1) the two states' metrical hierarchies are identical and (2) the time of the last tatum in S_{i-1} is equal to the time of the first tatum in S_i. Note that the second condition is invalid in the case of a transition from S_0 to S_1 since S_0 contains no tatums; in this case, we instead restrict $S_1.t_1$ to lie exactly on the first observed note's onset time.

The transition probability $P(S_i|S_{i-1})$ is shown in Equation (4.6), where the first term, defined in Equation (4.7), models the probability of any tempo change and the second term, defined in Equation (4.8), models the spacing of the tatums themselves.

$$P(S_i|S_{i-1}) = P(\mathrm{Tempo}(S_i)|\mathrm{Tempo}(S_{i-1}))P(S.t) \tag{4.6}$$

$$P(\mathrm{Tempo}(S_i)|\mathrm{Tempo}(S_{i-1})) = \begin{cases} \mathcal{N}_1(\mu_{t_0}, \sigma_{t_0}, \mathrm{Tempo}(S_i)) & i = 1 \\ \mathcal{N}_1\big(0, \sigma_t, \frac{\mathrm{Tempo}(S_i) - \mathrm{Tempo}(S_{i-1})}{\mathrm{Tempo}(S_{i-1})}\big) & i \geq 2 \end{cases} \tag{4.7}$$

$$P(S.t) = E(b \in S.t) \prod_{b \in S.t} \Big(E(sb \in b) \prod_{sb \in b} E(t \in sb)\Big) \tag{4.8}$$

In Equation (4.7), the tempo of the first bar (where $i = 1$) is assumed to be normally distributed around μ_{t_0} with standard deviation σ_{t_0}, while subsequent tempo changes

ing standard deviations, resulting in potentially wildly different results when evaluated at an identical number of standard deviations from the mean for different normal distributions. Additionally, since the distributions are used in contexts in which they cannot be properly normalised (due to their continuous domain), the precise probability value for $\mathcal{N}_1(\mu, \sigma, x)$ is calculated using a standard normal distribution with mean 0 and standard deviation 1 evaluated at $\frac{x-\mu}{\sigma}$.

are evaluated as the proportional change from the tempo of the previous bar, again normally distributed, this time with mean 0 and standard deviation σ_t.

For the tatum timings in Equation (4.8), the function $E(t)$, defined in Equation (4.9), evaluates the probability of the evenness of any given list of times. $E(b \in S.t)$ calculates this for all of the beats b in the state, while the terms $E(sb \in b)$ and $E(t \in sb)$ perform the same calculation for the sub beats in each beat and the tatums in each sub beat respectively.

$$E(t) = \begin{cases} \mathcal{N}_1(\mu_e, \sigma_e, \frac{\sigma(t)}{\mu(t)})/E_{norm} & \frac{\sigma(t)}{\mu(t)} \geq \mu_e \\ \mathcal{N}_1(\mu_e, \sigma_e, \mu_e)/E_{norm} & \frac{\sigma(t)}{\mu(t)} < \mu_e \end{cases} \tag{4.9}$$

$$E_{norm} = \frac{1}{2} + \frac{\mu_e}{\sigma_e}\mathcal{N}_1(\mu_e, \sigma_e, \mu_e) \tag{4.10}$$

$E(t)$ is a piecewise function which takes as input a list of the lengths of a group of tatums, sub beats, or beats (rather than their times). Here, $\mu(t)$ represents the mean of those lengths and $\sigma(t)$ represents the standard deviation of those lengths. The function is calculated as a modified normal distribution with mean μ_e and standard deviation σ_e, based on the input list's standard deviation as a proportion of its mean. If this proportion is greater than or equal to μ_e, the result is calculated from a straightforward normal distribution. Otherwise, the resulting value is exactly the value of a standard normal distribution evaluated at its mean.

This value is then normalised so as to ensure the new distribution's integral to again sum to 1 by dividing by the factor E_{norm}, defined in Equation (4.10) as the sum of two terms. $\frac{1}{2}$ is the area of the standard normal distribution greater than the mean, and $\frac{\mu_e}{\sigma_e}\mathcal{N}_1(\mu_e, \sigma_e, \mu_e)$ is the area of the rectangle formed by extending the peak of the standard normal distribution to the left until the value corresponding to 0 from the non-standardised normal distribution, as values less than this correspond to a negative $\sigma(t)$, which is not possible.

4.4.1.4 Optimisations

We use a modified Viterbi search to perform inference on our model, using a beam search where at each step we save only the b most probable hypothesis states (not including those still at S_0 with no tatums yet). As mentioned in Section 4.4.1.2, we use a supervisor to group notes into the appropriate note sets for each hypothesis state at each step. The supervisor is also used to reduce the search space by restricting the possible transitions from each state given the observed notes.

For the transition from S_0 to S_1, we introduce two heuristics: (1) the first tatum in S_1 must lie exactly on the onset of the first observed note and (2) the last tatum in S_1 must also lie exactly on a note onset, though which note specifically is not restricted by any means other than limiting the tempo of the first bar using t_{min} and t_{max}. According to these heuristics, for each S_0, the supervisor creates the observed note set for every possible S_1. Allowed times for the tatums in $S_1.t$ are also restricted based on each observed note set N_1. Essentially, all tatums are placed evenly unless there is a specific reason not to (i.e., unless a note onset lies close to a tatum).

Specifically, a given value for $S_1.t$ is legal if it can ever be generated by the following procedure. First, the appropriate number of beats (according to a given state's metrical hierarchy) are placed between the first and last tatum times, as if each tatum was evenly spaced. Next, each placed beat—excluding the last beat as well as the first—may be shifted to the location of any note whose onset time is within half of one sub beat length of the original beat location. Each beat (again excluding the first and last as appropriate) may then be nudged up to half of a tatum length around its location with a magnetism of M_b, as shown in Equation (4.11). Here, t is the original time of the beat, M is the magnetism (M_b in this case) which is used to control how far the beat is nudged, and N is the set of notes which lie within the given window. This equation can always return the original time, though it is also allowed to nudge the given time towards either the onset time of the closest note ($closest(N)$) or the average onset time of all notes within the window ($avg(N)$), if N is large enough. Sub beats are placed similarly: initially evenly between any of the existing beats, then nudged up to one tatum length around its location with magnetism M_{sb}. Notice that the sub beats are not shifted. Finally, tatums are placed evenly between the sub beats (and neither shifted nor nudged).

$$nudge(t,M,N) = \begin{cases} t & always \\ t + M(closest(N) - t) & |N| > 0 \\ t + M(avg(N) - t) & |N| > 1 \end{cases} \tag{4.11}$$

Allowed times for the tatums in $S_i.t$ for $i > 1$ are restricted to those which can be generated by the same procedure, with the exception that the final beat in $S_i.t$ may now be shifted and nudged. The initial beat locations are calculated such that with no shifting or nudging, $\text{Tempo}(S_{i-1}) = \text{Tempo}(S_i)$.

Even with the above restrictions, the search space is still large. As mentioned we use a beam search, where at each step we save only the top b most probable hypothesis states (not including those still at S_0 with no tatums yet). Before we remove those hypotheses which fall outside the beam, we remove hypotheses which are deemed to be too close to another more probable hypothesis based on a threshold Δ_{min}. Specifically, a hypothesis which has identical metrical hierarchy to a more probable hypothesis, and whose tempo and most recent tatum time both lie within Δ_{min} of that other hypothesis' tempo and most recent tatum time is removed.

In addition to the above search space reductions and speed optimisations, we describe two changes used to make our model more robust in regards to the idiosyncrasies of live performance such as staccato and ornamentation.

First, for handling staccato notes which are much shorter than their note values would suggest in the score, we extend each note's offset until either the onset of the following note in the bar within its voice (if one exists), or to the end of its bar. This allows the LPCFG, which is trained on metronomic MIDI where staccato is not present, to better recognise the rhythms present in live performance.

For handling ornamentation such as trills, we use a threshold $trill_{max}$. Any note whose onset time is within $trill_{max}$ of the onset time of the previous note within its voice is removed (though the removed notes are still used when deciding whether to remove the subsequent note). The overall effect of this process is that trills or any very fast ornamentation (which again would not be present in the LPCFG's training data) are reduced to a single short note with its onset at the start of the trill or ornamentation. If this optimisation is used in conjunction with the extend notes optimisation, the remaining notes are extended only after trills and ornamentation are removed, and the result is that a fast ornamentation is replaced by a single long note.

4.4.1.5 Example

An example of our model being run on the simple $\frac{2}{4}$ rhythm |𝅗𝅥|♩♩♪♪|𝅗𝅥| ... is shown in Figure 4.8. Most of the notes are played metronomically, except for the first eighth note whose onset is slightly early. Ground truth bars, beats, and sub beats are given by solid, dashed, and dotted vertical lines respectively. Four hypotheses are shown in rows, where the metrical hierarchy is displayed in the box on the left. $M_{b,s}$ refers to a hierarchy with b beats per bar and s sub beats per beat, l represents the number of tatums per sub beat, and a represents the anacrusis length. In each hypothesis state, tatums are shown by dots, ordered tatum, sub beat, beat, and bar from bottom to top. The supervisor is shown on the bottom observing every note individually, and each state observes a note set prepared by the supervisor at the appropriate time. Notice that the notes in the note sets are displayed by their ground truth value although their value in any given state may be different depending on the tatum locations.

The top hypothesis is a $M_{2,2}$ meter with 4 tatums per sub beat and an anacrusis length of 8 tatums. Initially, the supervisor lets this hypothesis observe the first half note at the time of the quarter note (so that it knows when to place the initial downbeat). S_1 for this hypothesis contains only nine tatums (since its anacrusis length is 8), and they are placed evenly. Following that, S_2 extends past the initial two bars shown, and would likely be given a relatively low $P(rhythm)$ due to its rhythm: an eighth note, followed by two sixteenth notes, followed by a quarter note. Also notice that its second sub beat does not match perfectly with the ground truth beat time, since it has been nudged, but not with a magnetism of 1. This hypothesis would correctly match most ground truth bars (as beats) and ground truth beats (as sub beats), and its probabilities would be relatively high except for perhaps $P(rhythm)$, since it gets two ground truth bars of notes for each of its bars.

The next hypothesis is a $M_{2,2}$ meter with 4 tatums per sub beat and no anacrusis, and correctly matches the ground truth meter. It initially observes the first half note, like the previous hypothesis, but since it has no anacrusis, places a full bar of tatums up to that note. Its S_2 is fully shown, after observing the notes from the second bar, and its tatums line up well with the ground truth. Its second beat has been shifted to exactly the correct location, and its third sub beat has been nudged very close to the ground truth location. This hypothesis would likely have the highest probability for our model given that the notes match up well with tatums, the rhythm and tempo both seem reasonable, and its tatums are relatively evenly spaced.

The next hypothesis is again a $M_{2,2}$ meter with 4 tatums per sub beat and no anacrusis; however, it first observes the first three notes together in a note set. Thus, in its S_1, though the first beat is shifted to the ground truth downbeat, therefore matching well, its second beat does not match the underlying notes, with the sub beat not being nudged enough to match the onset time of the first eighth note. Notice that in S_2, the tempo jumps back up to the average tempo from S_1, rather than the tempo of the last tatum in S_1. This hypothesis would not align its tatums well with the underlying notes and would therefore have a low probability.

The final hypothesis is a $M_{3,2}$ meter with 4 tatums per sub beat and no anacrusis, which initially observes a set of the first two notes. In S_1, its third beat shifts correctly and thus its tatums line up with the ground truth time well (except for the fact that it has the incorrect metrical hierarchy, with 3 beats per bar). In the beginning of its

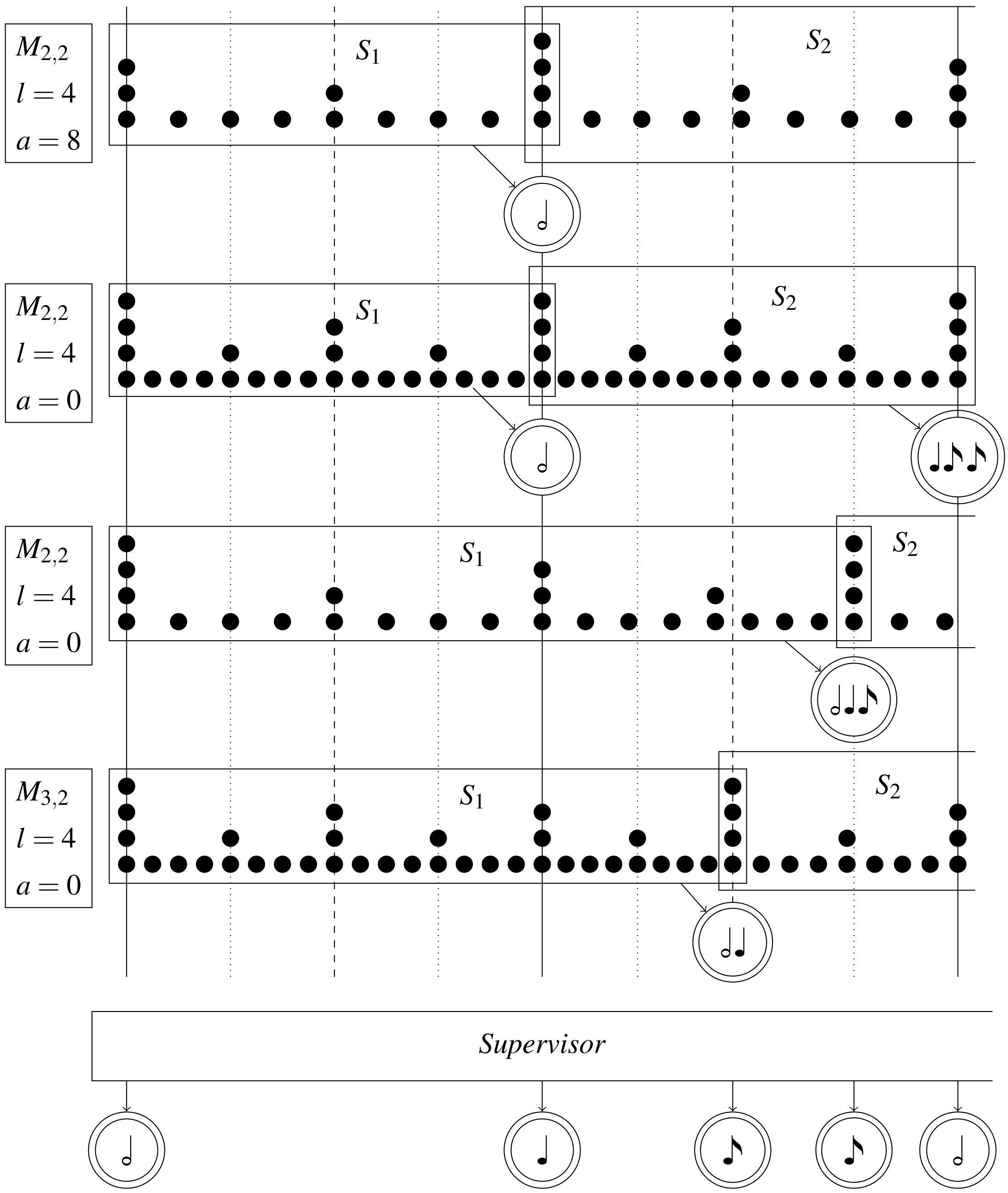


Figure 4.8: Four example hypotheses of our model being run on the simple $\frac{2}{4}$ rhythm |𝅗𝅥|♩♪♪|𝅗𝅥| .... Ground truth bars, beats, and sub beats are given by solid, dashed, and dotted vertical lines respectively. Four hypotheses are shown in rows, where the metrical hierarchy is displayed in the box on the left. $M_{b,s}$ refers to a hierarchy with b beats per bar and s sub beats per beat, l represents the number of tatums per sub beat, and a represents the anacrusis length. In each hypothesis state, tatums are shown by dots, ordered tatum, sub beat, beat, and bar from bottom to top. The supervisor is shown on the bottom observing every note individually, and each state observes a note set prepared by the supervisor at the appropriate time.

S_2, its second beat shifts to the correct location, and its second sub beat is nudged slightly towards the correct location as well, though not fully. This hypothesis would match most beats and sub beats correctly, but miss the bars. It's evenness and tempo probabilities would be high, and the notes match up well with its tatum times, but its $P(rhythm)$ would be low due to its odd bar timings.

Given this simple example, and only looking at 4 hypotheses, our model would assign the correct hypothesis the greatest probability. In practice, there would be many more meter types and anacrusis lengths, and the supervisor would create more S_1s with a wider variety of initial note sets for each hypothesis. However, this simple example serves well to show our model in action and to hopefully make its description more clear.

4.4.2 Evaluation

4.4.2.1 Corpora

For evaluation, we use two corpora: one containing metronomic MIDI files of the 48 fugues from Bach's Well Tempered Clavier (WTC)[3] and Bach's 15 Inventions[4], and another of 13 live performance MIDI files of Bach's fugues and preludes from the WTC, taken from CrestMusePEDB[5] (Hashida et al., 2008). For training, we use an additional corpus: the miscellaneous corpus, also used by Temperley (2009) for training, divided into a live performance portion and a metronomic portion.

Since our model requires voice assignments, we run the model from Section 3.2 as a preprocessing step for all corpora.

4.4.2.2 Training

To train most of the parameters for the beat tracking model, we measure statistics from the live performance portion of the miscellaneous corpus. To set the mean and standard deviation of the initial bar's tempo, μ_{t_0} and σ_{t_0}, we measure the mean and standard deviation of the tempi of the first bars of all of the pieces. The standard deviation of tempo changes, σ_t is set similarly, measuring the standard deviation of the proportional change in tempo between each consecutive bar in the pieces. The evenness mean and standard deviation, μ_e and σ_e, are set as follows. For each bar in the

[3] The fugues were acquired from `www.musedata.org`.

[4] The inventions were acquired from `www.imslp.org`.

[5] We do not include the 13th prelude from WTC Book I due to an error in the file.

training corpus, we measure its evenness as the standard deviation of the beat lengths in the bar divided by the mean of the beat lengths in that bar. Then, μ_e and σ_e are set to the mean and standard deviation of these evenness values throughout the entire corpus. (Even though these values are used for all levels of the metrical structure in our model, the annotated beats in the corpus are the most reliable, and we make an assumption that the evenness should be relatively constant throughout all metrical levels.) The note standard deviation σ_n is set by measuring the standard deviation of the difference between each note and the closest tatum throughout the pieces. The initial bar tempo bounds t_{min} and t_{max} are set by measuring the minimum and maximum tempi of any initial bar throughout the pieces, and rounding to the nearest 100000 μs.

The remaining parameters are set in an ad hoc fashion through testing on the miscellaneous corpus, and we have found our model's performance not to be very sensitive to the precise values used. We use a magnetism of 1.0 at the beat level (M_b) and 0.5 at the sub beat level (M_{sb}) so that the beats are more likely to lie precisely on a note (as beats are more salient than sub beats). We take 0.1 s to be $trill_{max}$, the maximum length for a note to be considered ornamentation, set by looking at typical ornamentation note lengths in the corpus. Δ_{min} and b are simply optimisations used to improve the speed of our model, and we use values of 1 ms and 200 respectively, though in practice, lower values of Δ_{min} or higher values of b only improve our model's performance.

All of the parameter values for our beat tracking model used for evaluation are listed in Table 4.4.

For our standard evaluation, we train the LPCFG's probabilities from the metronomic portion of the miscellaneous corpus, since this allows for a direct comparison with the model of Temperley (2009). However, as we discussed in Section 4.3, the grammar is sensitive to a lack of training data, particularly a lack of training data in the style of the evaluation corpus, which happens when training on the miscellaneous corpus for evaluation on Bach compositions. To investigate this further, we also run experiments when training the LPCFG's probabilities on a superset containing the metronomic portion of the miscellaneous corpus as well as the entire metronomic corpus of Bach compositions. Note that when evaluating this version of our model, we leave out the piece currently being evaluated from the grammar's training set so as to avoid overfitting. In all experiments, we train the LPCFG with data that has undergone the same optimisations as the data to be evaluated (in terms of extending notes and removing trills and ornamentation).

Parameter	Value
μ_{t_0}	1.088500 s
σ_{t_0}	709.918 ms
σ_t	0.0743
μ_e	0.0181
σ_e	0.0336
σ_n	6.655 ms
t_{min}	0.4 s
t_{max}	3 s
M_b	1.0
M_{sb}	0.5
$trill_{max}$	0.1 s
Δ_{min}	1 ms
b	200

Table 4.4: The parameter settings we use for our model's evaluation.

4.4.2.3 Metric

To evaluate our model's performance, we want a metric that is able to capture partial correctness, for example the misclassification of beats as bars or the grouping of bars together. Many existing beat tracking evaluation metrics take into account only detected beats, plus some classification of the time signature, which would mark such metrical hypotheses as incorrect. We would like a metric similar to that from Section 4.3.2.1; however, as it is designed for use mainly on beat-aligned data where a metrical hypothesis can not move in and out of phase throughout a piece, a few adjustments must be made to adapt it for use on live performance data.

We call this newly designed evaluation metric the metrical F-measure. It takes into account every grouping at three levels of the metrical hierarchy throughout an entire piece: the sub beat level, the beat level, and the bar level. For each hypothesised grouping at these metrical levels, we check if it matches a ground truth grouping. A hypothesised grouping is said to match a ground truth grouping if its beginning and ending times are each within 70 *ms* of the beginning and ending times of the ground truth grouping, regardless of the metrical level of either grouping.[6] Each matched pair of groupings within a piece count as a true positive, while any unmatched hypothesis

[6] This 70 *ms* window is taken directly from a popular beat tracking metric (Dixon, 2007).

groupings count as false positives, and any unmatched ground truth groupings count as false negatives. The metrical F-measure of a piece is then calculated as the harmonic mean of precision and recall as usual, and our reported results average these metrical F-measures across all songs in each corpus. Because the resulting scores are not normally distributed, statistical significance is calculated using a Wilcoxon signed rank test.

4.4.2.4 Results

We compare our model against that of Temperley (2009), which is trained entirely on the miscellaneous corpus. However, Temperley's model does not explicitly identify bar lines. Rather, it identifies beats and a single level above, consisting of either two or three beats grouped together, and it therefore cannot explicitly model both the beat level and the bar level of a piece with four beats per bar. This is a slight drawback to the model, and we report two scores for it: one taking the level above the beats to be bars in all cases, and another using a bar-score optimisation as follows. For pieces with four beats per bar, where Temperley's model has grouped beats into groups of two, we assign the maximum metrical F-measure of three different bar settings: (1) grouping the beats into bars with two beats per bar as given by the model, (2) grouping the beats into four beats per bar starting at the first complete beat grouping given by the model, and (3) grouping the beats into four beats per bar starting at the second complete beat grouping given by the model. This bar-score optimisation essentially results in an upper bound for Temperley's model's performance, assuming it makes the best possible bar level choice for all pieces with four beats per bar, and we mark with an * wherever it is used.

For direct comparison, the standard version of our model is trained on the same corpus as Temperley's model, but we present an evaluation of a few different versions of it based on different optimisations or training data. Results can be found in Table 4.5, where +T indicates use of the remove trills and ornamentation optimisation, +X indicates use of the extend notes optimisation, and +Bach indicates that the LPCFG training was augmented with the additional Bach compositions. We do not also augment Temperley's model with additional training data because there is no straightforward way to do so, and the model does not seem to be one which would be as sensitive to a lack of training data as our model.

The results show that on metronomic data, our model without optimisations significantly outperforms Temperley's when using identical training data ($p < .05$). The optimisations offer no significant improvement (which is unsurprising as they were de-

Method	Metronomic	Live Performance
Temperley (2009)	67.65	47.62
Temperley (2009)*	69.45	49.84
This Work	78.71	39.63
+T	75.36	39.40
+X	79.89	45.27
+X +T	77.67	47.81
+Bach	80.48	38.21
+Bach +T	80.08	42.35
+Bach +X	**80.50**	55.43
+Bach +X +T	80.48	**56.51**

Table 4.5: The average metrical F-measure of our method compared against those of Temperley (2009) on our two corpora. The * indicates Temperley's model's performance using the bar-score optimisation. For our model, +T indicates use of the remove trills and ornamentation optimisation, +X indicates use of the extend notes optimisation, and +Bach indicates using the additional Bach training data for the LPCFG.

signed specifically to help with live performance), but augmented training data leads to a small (not statistically significant) but consistent increase in performance across all optimisation configurations. On live performance, our model without optimisations underperforms Temperley's. However, the optimisations lead to increased performance: our model using both optimisations matches Temperley's non-optimised performance with identical training data, and exceeds it by almost 9 points with augmented training data (though this difference is not statistically significant). The effect of each of our model's optimisations is discussed in detail in Section 4.4.2.4.1.

The distribution of metrical F-measures for Temperley's model, run on live performance data, appears to be binomial: of the 13 pieces, three score below 20, while six score above 55, indicating that while the model performs well in general, it sometimes guesses a meter which is nearly entirely incorrect. With both optimisations, on the other hand, our model's scores are normally distributed around 65, with 8 pieces scoring between 55 and 75. Additionally, no pieces score below 20, indicating that it is more likely to make some partially correct guess, even if it is not entirely correct. The 1st prelude from WTC Book I illustrates this difference in performance, and its first bar is shown in Figure 4.9 along with the results of Temperley's model and our

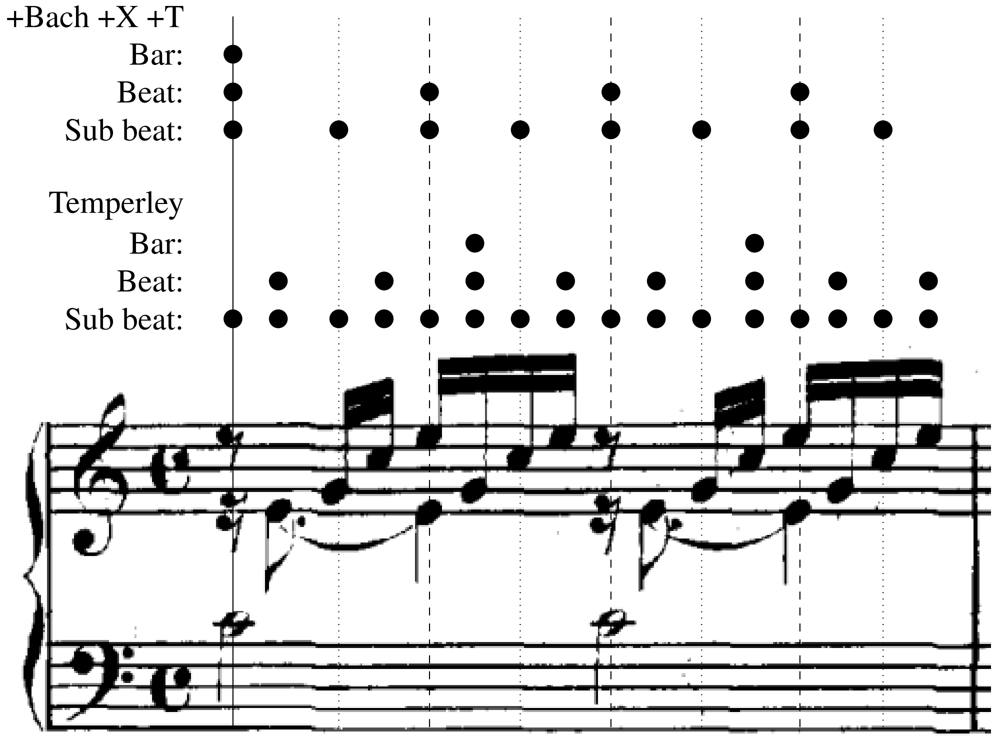


Figure 4.9: The first bar of the 1st prelude from WTC Book I (BWV 846). Above the music, the results from Temperley's model (bottom) are shown as well as the results from our +Bach +X +T model (top). Here, a dot represents a tatum placement at either the bar, beat, or sub beat level. The ground truth bar is given by the solid vertical line, ground truth beats are given by dashed vertical lines, and ground truth sub beats are given by dotted vertical lines.

+Bach +X +T model. The piece is in $\frac{4}{4}$ time, and Temperley's model achieves a score of only 15.74, guessing a $\frac{3}{8}$ time whose beats are even out of phase with the ground truth sub beats throughout much of the piece. On the other hand, our model scores 93.27, guessing a $\frac{4}{4}$ time which begins perfectly aligned, although it does shift slightly out of phase later in the piece.

One example of a piece for which Temperley's model outperforms ours is the 2nd prelude of WTC Book II, the first bar of which is shown in Figure 4.10 along with the results from Temperley's bar-score optimised model and our +Bach +X +T model. For this piece, Temperley's model achieves a score of 83.30 while ours only manages a score of 61.83. This piece is in $\frac{4}{4}$ time and contains relatively non-syncopated rhythms, with many bars containing either only sixteenth notes or only eighth notes in a given voice, as can be seen in the figure. While Temperley's model captures this meter

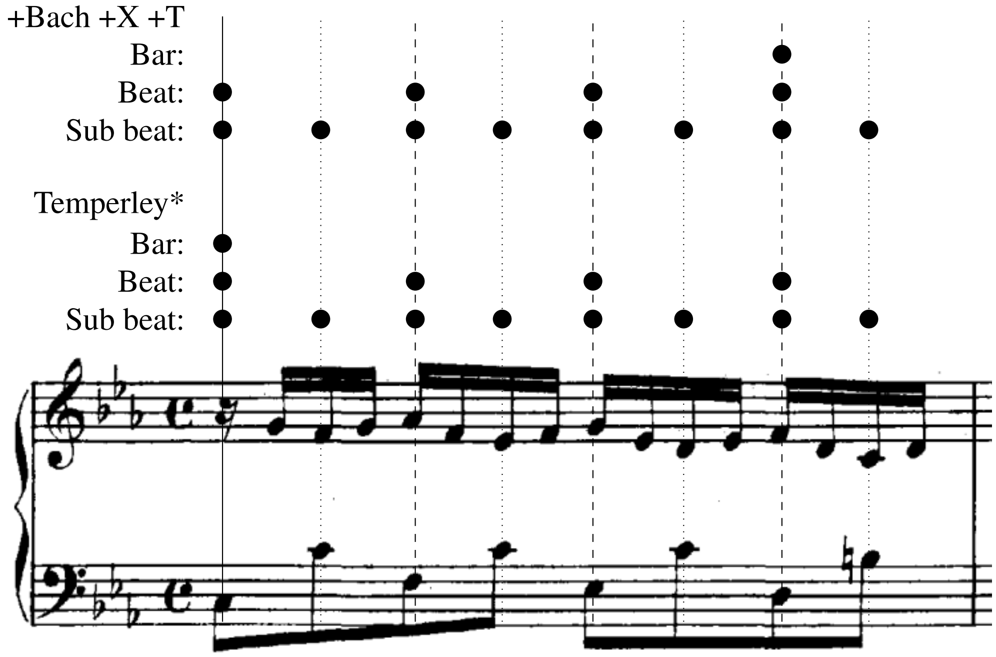


Figure 4.10: The first bar of the 2nd prelude from WTC Book II (BWV 871), showing an example the nearly isochronous bars which give our model problems. Above the music, the results from Temperley's bar-score optimised model (bottom) are shown as well as the results from our +Bach +X +T model (top). Here, a dot represents a tatum placement at either the bar, beat, or sub beat level. Ground truth bars are given by solid vertical lines, ground truth beats are given by dashed vertical lines, and ground truth sub beats are given by dotted vertical lines.

correctly (with some phase errors), our model guesses a $\frac{4}{4}$ time which is early by a single beat. Our model has some difficulty finding the correct phase of isochronous melodies since it uses no pitch or harmonic information (which are the most salient indicators of metrical phase in such isochronous pieces). Temperley's model, on the other hand, also includes chord detection, allowing it to better handle such melodies.

4.4.2.4.1 Optimisations Another aspect of our model to investigate is the effect of the different optimisations on its performance. As can be seen from Table 4.5, they (+X and +T) have little effect on metronomic data (which is not surprising given that they are designed specifically for live performance). However, on live performance data, they do indeed improve performance. Both with and without augmented training data, the remove trills optimisation has a small effect by itself (essentially none without the data and very small with it, and not statistically significant in either case), but

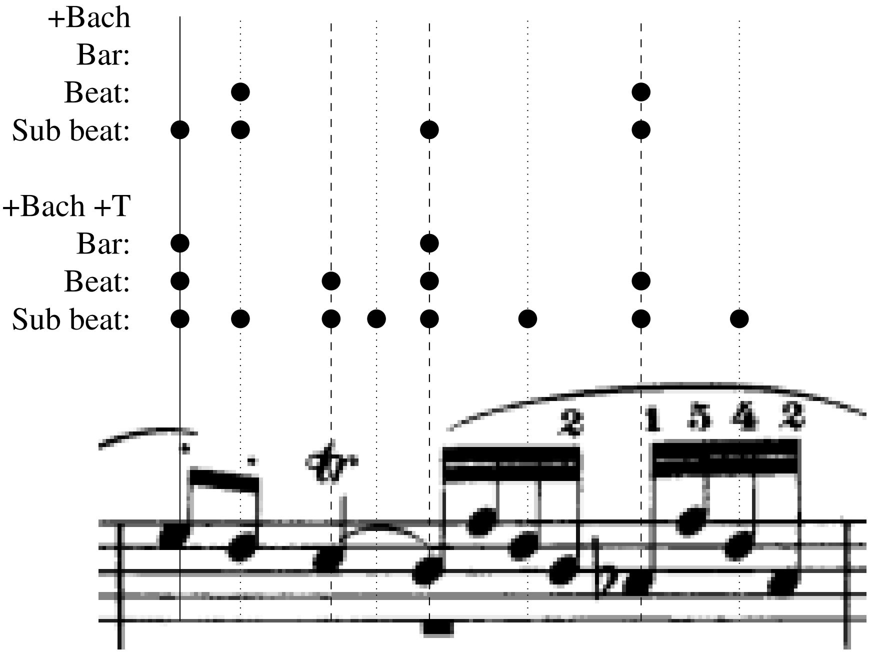


Figure 4.11: The treble clef of the second bar of the 7th fugue from WTC Book I (BWV 852). The trill on the second beat of this excerpt repeats throughout the piece, leading the +Bach model to lengthen its beat length such that the trill is interpreted as 16th notes. Above the music, the results from our +Bach +T model (bottom) are shown as well as the results from our +Bach model (top). Here, a dot represents a tatum placement at either the bar, beat, or sub beat level. The ground truth bar is given by a solid vertical line, ground truth beats are given by dashed vertical lines, and ground truth sub beats are given by dotted vertical lines.

extending notes leads to a significant improvement ($p < .05$). The combination of both optimisations improves performance even further (though not statistically significant), leading to peak performance both with and without augmented training data.

One specific example where the remove trills optimisation leads to improvement with augmented training data is on the 7th fugue from WTC Book I. The treble clef of the second bar of the piece is shown in Figure 4.11, including the results from our +Bach and +Bach +T models, which achieve scores of 31.58 and 60.20 respectively. The trill on the second beat of this excerpt repeats throughout the piece, leading the +Bach model to lengthen its beat length such that the trill is interpreted as 16th notes. With the remove trills optimisation, however, our model is able to find the correct metrical structure. Essentially, the remove trills optimisation frees our model from the constraint of trying to align its tatums with all of the notes in a trill or ornamentation.

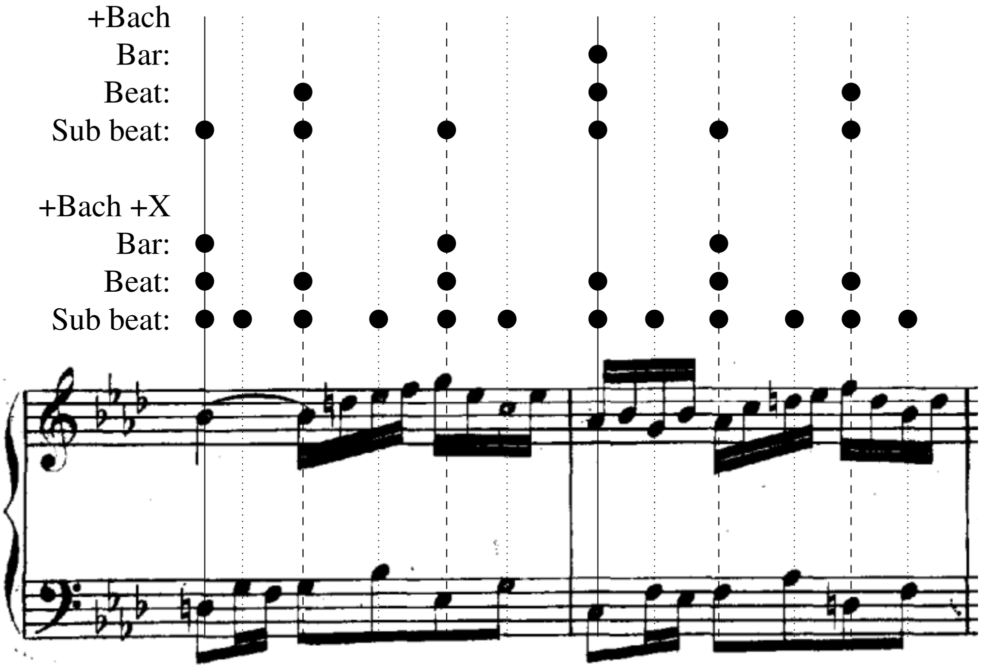


Figure 4.12: Bars 10 and 11 of the 17th prelude from WTC Book I (BWV 862), showing the pattern which repeats throughout the piece in the bass voice. All of the eighth notes in the pattern are played staccato, something which our extend notes optimisation is able to overcome. Above the music, the results from our +Bach +X model (bottom) are shown as well as the results from our +Bach model (top). Here, a dot represents a tatum placement at either the bar, beat, or sub beat level. Ground truth bars are given by solid vertical lines, ground truth beats are given by dashed vertical lines, and ground truth sub beats are given by dotted vertical lines.

An example of a piece for which the extend notes optimisation makes an improvement is the 17th prelude from WTC Book I. In this piece, in $\frac{3}{4}$ time, the lowest voice has a very common repeated rhythm of an eighth note followed by two sixteenth notes followed by four more eighth notes, where the eighth notes are all played staccato. Bars 10 and 11 of this piece, which contain the pattern, are shown in Figure 4.12 along with the results from our +Bach model with and without the extend notes optimisation. With the optimisation, our model correctly recognises the beat and sub beat levels, although it incorrectly guesses $\frac{2}{4}$ time rather than the correct $\frac{3}{4}$ time, scoring 53.59. Without the optimisation, on the other hand, these eighth notes are not as salient, and the model instead guesses a $\frac{2}{2}$ meter which moves in and out of phase throughout the piece, achieving a score of only 16.47. Throughout the corpus, the extend notes optimisation helps find strong notes whenever they are played staccato.

The combination of both optimisations improves overall performance even further, enabling the model to handle both staccato passages and ornamentation. The improvements from both optimisations are seen in the fully optimised model, alongside other slight improvements throughout the corpus such as fixing the placement of a single misaligned beat here or there. For example, in the previously discussed 17th prelude from WTC Book I (see Figure 4.12), the fully optimised model achieves a metrical F-measure of 60.35 while no other model eclipses a score of 54, even though the basic metrical alignment (a $\frac{2}{4}$ meter) does not change between the +Bach +X model and the fully optimised one.

4.5 Conclusion

In this chapter, we have described models for metrical structure detection and alignment from MIDI data, both metronomic and live performance.

First, we have presented an LPCFG for full metrical structure detection and alignment of metronomic data, and we have shown that our grammar improves over multiple baseline methods when run on metronomic MIDI data using a novel metric proposed for the task. The fact that lexicalisation adds definite value over a standard PCFG shows that there are complex rhythmic dependencies in music which such lexicalisation is able to capture.

We have also presented an HMM which performs metrical structure detection and alignment on live performance data, and we have shown that the model achieves state-of-the-art performance when evaluated on a corpus of metronomic data, as well as a second corpus of live performance data. The HMM incorporates the LPCFG as one component, which allows both models to work with each other to align an input piece with a detected metrical structure. This joint model requires no information a priori except for note onset and offset times, and is both probabilistic and incremental. We have also proposed a new metric for the task, which takes into account vertical misalignments (for example, those which align the beat level of a piece with bars).

Metrical structure detection and alignment are clearly important tasks for any complete transcription system, and we have shown that our joint model is able to perform them well, even using only rhythmic data. Incorporating additional information such as pitch or harmony should only lead to better performance. Specifically, it has been shown that harmonic changes are most likely to occur at the beginnings of bars (Papadopoulos & Peeters, 2011), and that low notes may be a salient feature of strong beats in addition to note duration (Dixon, 2001). In fact, without such additions, our grammar cannot hypothesise a full meter for a perfectly isochronous melody.

The LPCFG itself is somewhat sensitive to a lack of training data, though it does learn metrical stress patterns quickly, and we have shown that performance continues to improve as more training data is given to it. More aggressive cross-level back-off techniques could also in theory make the grammar more robust to such a lack of data. For example, it may be possible to model the transitions at the sub beat level of a $\frac{9}{X}$ meter type using the beat level transitions of a $\frac{3}{X}$ meter type. Similar cross-level learning techniques could also allow our grammar to be applied to more uncommon or irregular meter types such as $\frac{5}{X}$ or $\frac{7}{X}$, perhaps as the concatenation of the more common meter types.

Chapter 5

Joint Analysis

This chapter presents work towards the goal of a joint music language model consisting of voice separation (see Chapter 3), metrical structure detection and alignment (see Chapter 4), note value detection (not covered), and harmonic analysis (not covered). Thus far, we have only run and evaluated each aspect of our model independently. A joint analysis would bring our proposed model one step closer to a complete music language model for AMT.

Section 5.2 first presents the joint model, a combination of the voice separation model from Chapter 3 and the metrical analysis model from Chapter 4, and includes a discussion on different strategies for their integration. For the evaluation of such a joint model, there is of course the option of evaluating each aspect of the model independently, which we do. However, as we are trying to move towards a joint model, a joint evaluation strategy would be preferable. To that end, Section 5.3 contains a discussion on AMT evaluation, and proposes a metric for the joint evaluation of complete AMT systems, as well as strategies and principles that allow it to be adapted for use with models that perform only certain subtasks of a complete transcription, such as our joint model.

This chapter is based on the published work "Evaluating automatic polyphonic music transcription" (McLeod & Steedman, 2018a).

5.1 Introduction

A music language model which analyses many different aspects of music jointly is preferred, as it has been shown it previous work that different aspects of a musical analysis can inform each other. For example, Papadopoulos and Peeters (2011) showed that chord changes tend to align with downbeats (and it is not unreasonable to hypothesise that such a claim can be generalised to stating that chord changes tend to occur in stressed positions in a piece's metrical structure). We also noted in Chapter 3 that some knowledge of metrical structure could improve voice separation performance. Furthermore, our experiments in Section 3.3 showed that a joint approach to multi-pitch estimation and voice separation model outperforms a combination of identical components which is instead run independently sequence.

Certainly there is a benefit to performing joint analysis, but there is also a cost. The conceptual complexity of two models blows up significantly when they are combined into one. Nevertheless, we present such a model here in Section 5.2. Specifically, we combine our voice separation model from Chapter 3 and our metrical analysis model from Chapter 4 into a single joint probabilistic model, which runs incrementally and requires only note onsets, offset, and pitches as input. We evaluate our model against other simpler, non-integrated versions of our two models.

In Section 5.3, we discuss the joint evaluation of music transcription and analysis, and propose a new metric for the task. Our metric is designed in a way that it does not penalise a model twice for a single error; for example, it will not assign a penalty for not assigning a note to the correct voice when the model has not even detected that note in the first place. We also show our metric's effectiveness through the use of specific examples, and present an evaluation of our own model using it. While our joint model does not perform a full transcription and analysis from audio data (and to our knowledge no model yet exists which does), our metric is modularised in a way that it can be applied to any subset of the tasks involved.

5.2 Joint Model

Integrating our voice separation model with our metrical alignment model is a relatively simple process, since both were designed with such an integration in mind. Both models are HMMs which are incremental, probabilistic, and require no a priori information. Thus, our joint model can be created intuitively by simply defining each joint state to be the combination of a state from the voice separation HMM and a state from the metrical alignment HMM. The transition and emission probabilities of the joint HMM are simply calculated as the product of the transition and emission probabilities of each individual HMM.

A complication arises when actually performing inference on the joint HMM. We again use a modified Viterbi search with a beam, but in order to handle the complexity of the state space, our integrated beam size should be at least the product of the beam size used for voice separation alone (25), and the beam size used for metrical alignment alone (200). The resulting beam size of 50000 is too large to be reasonably tractable. Thus, we use a smaller beam size b along with two additional constraints.

First, we do not remove a hypothesis state from our search unless it has observed at least a full bar of notes at its tempo. This prevents our beam from filling up with hypotheses with very fast tempi without giving the slower hypotheses a chance to begin. Second, we introduce a voice beam with size b_v, which can override the first constraint, and prevents the beam from filling up with too many different voice separation hypotheses (since the required beam for voice separation is much smaller than that for metrical alignment). Specifically, if we have greater than b hypotheses in the beam (due to the first constraint), and those hypotheses consist of more than b_v unique voice separation hypotheses, we repeatedly remove the least probable joint state associated with the least probable voice separation state from the beam until either (1) we have only b hypotheses left in the beam, or (2) we have only b_v unique voice separation hypotheses in the beam.

5.2.1 Results

We evaluate our joint model on 13 live performance MIDI files of Bach's fugues and preludes from the WTC, taken from CrestMusePEDB[1] (Hashida et al., 2008). This is the same live performance corpus used for evaluation in Chapter 4. For the voice separation model's parameters, we use those learned for evaluating the Live WTC corpus in Chapter 3. For the metrical alignment model, we use the version with both optimisations and augmented training data, and we use the same parameter settings noted in Chapter 4. For the integrated model, we use a beam size of $b = 1000$ and a voice beam size of $b_v = 5$.

To evaluate our joint model, we present both voice separation results (using the F-measure described in Section 3.2.3.3) and metrical alignment results (using the metrical F-measure proposed in Section 4.4.2.3). We compare our integrated model against two baselines. First, our two individual HMMs, run entirely independently (the numbers are taken from their respective individual evaluations in Chapters 3 and 4). Second, the two HMMs run sequentially as follows. We first run the voice separation

[1] We do not include the 13th prelude from WTC Book I due to an error in the file.

Model	Voice	Meter
Independent	94.34	56.51
Sequential	**94.56**	**57.33**
Joint	85.78	27.77

Table 5.1: The results of our joint model on both voice separation and metrical alignment compared against those of our two baselines: Independent, which runs each individual model entirely independently; and Sequential, which runs both individual models in sequence rather than simultaneously.

HMM with the same parameter settings as above, except with a beam size of 5. That is followed by running the metrical alignment model with the same parameter settings as in Chapter 4 on each of the 5 voice separation hypotheses. The most probable joint state is formed as the one with the greatest combined probability (taken as the product of the final voice separation state's probability and the associated final metrical alignment state's probability).

The results are summarised in Table 5.1, where it can be seen that the sequential version of our joint model achieves the best results, outperforming the baseline of each component running entirely independently. This shows that the two components of the model are indeed able to work together to find a more accurate analysis. The fully joint model, however, clearly suffers from too small of a beam, performing substantially worse than the other two models. It seems that a more sophisticated integration strategy would be required in order for such a model to be feasible. In particular, it tends to align each piece with a slow $^{12}_{4}$ meter, indicating that a larger beam (or a more sophisticated search modification) would be required for increased accuracy. Essentially, because of the large branching factor of our joint model, these slow meters are able to fill up the beam rapidly, and because they do not assign their metrical alignment a probability until the end of a bar, by the time such a probability calculation is made, the beam is already full with these slow meters. Both the placement of beats and the structure of a rhythmic tree in our LPCFG depends strongly on a bar's entire context, so waiting until the end of each bar to assign a hypothesis a probability is unfortunately necessary. It is possible that a rough probability estimate (or an upper bound, as in A* search) could be calculated for the incomplete trees, but we leave the calculation and integration of such estimates for future work.

Most pieces from the corpus see very little change for the sequential model com-

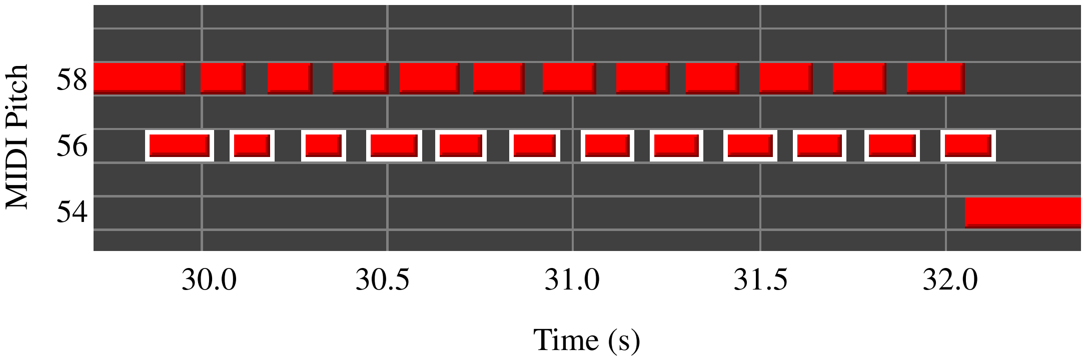


Figure 5.1: A MIDI representation of one instance of the repeated trill from the 23rd fugue from WTC Book I (BWV 868). Here, all of the notes belong to the second-lowest voice. The notes shown with a white border are erroneously assigned to the lowest voice—which has no notes at the time—by the independent model, while the sequential model makes the correct voice assignments.

pared with the independent model in voice separation accuracy, with largest increase in performance coming from the 23rd fugue from WTC Book I, whose F-measure increases from 89.81 to 92.32. It has a repeated trill which occurs three times throughout the piece, and a MIDI representation of one of these occurrences is shown in Figure 5.1. Run independently, the voice separation model assigns the bottom note of the trill to a different voice than the top note of the trill every time. In the figure, this results in the white-bordered notes being assigned erroneously to the lowest voice, while the other notes are assigned correctly to the second-lowest voice. In particular, since the notes tend to overlap each other, the probability of assigning them all to the same voice is slightly lower than this multi-voice assignment, all else being equal. With the metrical alignment model, however, the rhythm resulting from the multi-voice assignment (repeated short notes which do not directly line up with any particularly probable metrical grid) is assigned a very low probability. Instead, the sequential model prefers to correctly assign all of the notes of each trill to a single voice. The metrical alignment model then recognises the pattern as a trill and removes the trill notes before its rhythmic probability calculation, resulting in a rhythm with a much greater probability. Thus, although this single-voice assignment has a slightly lower voice assignment probability, the combined probability is greater, and the sequential model makes the correct voice assignment.

The metrical F-measures change more often than the voice separation F-measures in the sequential model compared to the independent model, with some of the values

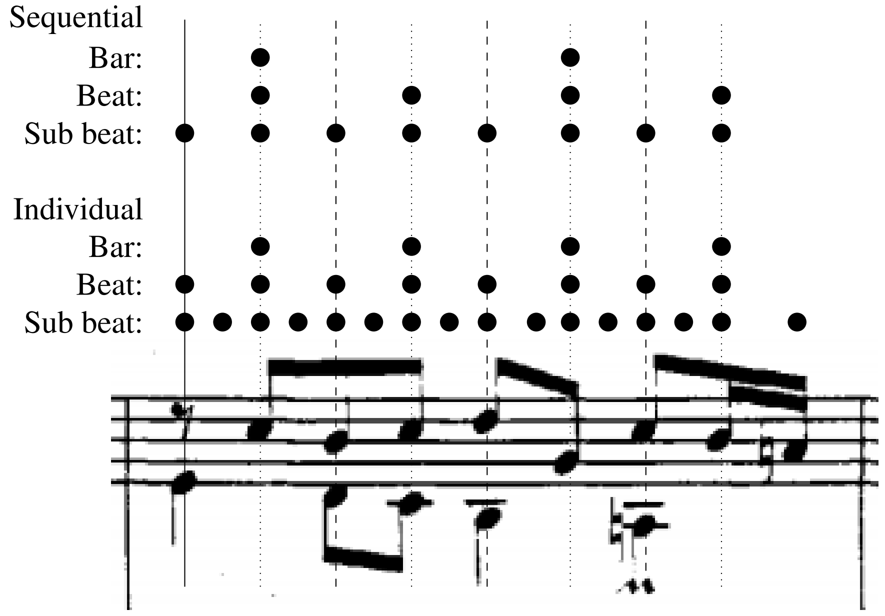


Figure 5.2: The second bar of the 2nd fugue from WTC Book II (BWV 871). Above the music, the results from our individual model (bottom) are shown as well as the results from our sequential model (top). Here, a dot represents a tatum placement at either the bar, beat, or sub beat level. Ground truth bars are given by solid vertical lines, ground truth beats are given by dashed vertical lines, and ground truth sub beats are given by dotted vertical lines.

decreasing (due to more noisy voice separation results), and some increasing. The most substantial increase comes from the 2nd fugue from WTC Book II, the second bar of which is shown in Figure 5.2. This piece is in $\frac{4}{4}$ time, and while the individual version of the model guesses an out-of-phase $\frac{2}{8}$ time, the sequential version instead guesses out-of-phase $\frac{2}{4}$ time. While the sequential version is still equally out-of-phase, it has aligned its sub beats (rather than its beats, as in the independent model) with the ground truth sub beats. Thus, its false positives come from its beats and bars (approximately six per ground truth bar), rather than the independent model, which has false positives much more often from its sub beats and bars (approximately 20 per ground truth bar). The sequential version therefore has a much higher precision, while maintaining its recall, resulting in a higher metrical F-measure than the independent version of our model.

It appears that the sequential model improves over the independent model in metrical F-measure because it is able to take a note or two which, in the correct voice, would result in an improbable rhythm according to the grammar, and instead assign them to a different voice, leading to a slightly more likely rhythm. This ability enables the sequential model to improve its metrical F-measure at the expense of a very small decrease in voice separation performance (which is overshadowed by the increase in voice separation performance described above). We would expect that the fully joint model would continue to improve over the sequential model, were its beam size not a hindrance.

5.3 Joint Evaluation

In this section, we propose a new metric to quantitatively evaluate AMT systems that perform both multi-pitch detection and a complete music analysis including voice separation, metrical alignment, note value detection, and harmonic analysis. The metric, *MV2H* (from **M**ulti-pitch detection, **V**oice separation, **M**etrical alignment, note **V**alue detection, and **H**armonic analysis), can be easily adapted for systems that do not perform all of the tasks involved, such as our joint model from the previous section, and Section 5.3.3 includes an analysis of our model's performance using the metric. The code for the evaluation metric described here is available at https://www.github.com/apmcleod/MV2H.

One of the main principles of the new metric is that of disjoint penalties: that mistakes should only be penalised once. That is, if an error in one part of the transcription causes a mistake in another part, that error should only be counted once. For example, if a pitch is missed during multi-pitch detection, the metric should not further penalise missing that note from the voice separation results.

Based on this principle, we do not include errors related to the proper typesetting of a transcription in our metric, and we do not even require a typeset musical score to perform our evaluation. Most typesetting decisions come down to the proper analysis of the underlying piece. For example, if metrical alignment is performed properly, beaming comes naturally. Likewise, stem directions can follow from voice separation and pitch spelling is a consequence of a proper harmonic analysis. For details related to the proper typesetting of music and its relation to the underlying music analysis, see Gould (2011).

5.3.1 Existing Metrics

Each of the separate tasks involved in the full AMT process has been the subject of much prior research, and there are existing metrics for each of them. This section gives a brief overview of the most widely used metrics for each subtask.

5.3.1.1 Multi-pitch Detection

Multi-pitch detection is evaluated both at the frame level and at the note level depending whether a given model includes some form of note tracking or not (see Chapter 2 for an overview of existing multi-pitch detection systems). However, as the goal of this section is to define a metric which is useful for a complete AMT system, the note-level evaluation metrics are most applicable here. Readers interested in the frame-based evaluation, an accuracy metric, should refer to Poliner and Ellis (2007), where it was first introduced.

For the note-level metric, a note is defined by its pitch, onset time, and offset time. Bay et al. (2009) define two different precision, recall, and F-measures for note-level multi-pitch detection. For the first, they define true positives as those notes detected whose pitch lies within a quartertone (3%) of that of a ground truth note, and whose onset time is within 50 ms of the same ground truth note's onset time, regardless of offset time. Spuriously detected notes are each assigned a false positive, and ground truth notes which are not matched by a detected note are each assigned a false negative. The second metric they propose is identical, with the additional constraint that a detected note's offset must be accurate to within 20% of the corresponding ground truth note's duration, or lie within 50 ms, whichever is larger, for it to be considered a true positive. Both of these metrics are used in the Music Information Retrieval Evaluation Exchange (MIREX) Multiple Fundamental Frequency Estimation & Tracking Task (MIREX, 2017e), and in the mir_eval package by Raffel et al. (2014).

For our purposes, we care mostly about onset time and pitch (to the nearest semitone) as these aspects are most directly relevant to the underlying musical score. Offset time, on the other hand, is applicable as far as it relates to note value, and is discussed in Section 5.3.1.4. Our multi-pitch detection metric will therefore be based most closely on the first multi-pitch F-measure, which doesn't account for offset time.

5.3.1.2 Voice Separation

In addition to AVC and the simple F-measure, both of which are detailed in Section 3.2.3.3, Kirlin and Utgoff (2005) define two metrics: *soundness*, which measures the percentage of consecutive notes in an assigned voice which belong to the same ground truth voice; and *completeness*, which measures the percentage of consecutive notes in a ground truth voice which were assigned to the same voice.

Each of these metrics would penalise an AMT system for any spurious notes, so for our proposed metric, we will need to use a modified version of one of them (or design a new metric) in order to enforce the principle of disjoint penalties.

5.3.1.3 Metrical Analysis

Metrical analysis is most often approached as one of three different tasks: downbeat tracking, beat tracking, or metrical structure detection and alignment. Downbeat tracking and beat tracking each involve identifying points in time, and thus can theoretically be evaluated using the same metrics, which are summarised by Davies, Degara, and Plumbley (2009), and more recently by Davies and Böck (2014). F-measure (which downbeat tracking work uses almost exclusively), described by Dixon (2007), is calculated by counting the number of (down)beats which lie within some window length (usually 70 ms) of an annotated (down)beat, and is used in the MIREX Audio Downbeat Estimation task (MIREX, 2017c). Cemgil, Kappen, Desain, and Honing (2000) propose a similar metric for beat tracking, where accuracy is calculated instead using a Gaussian window around each annotated beat. Goto and Muraoka (1997) propose a binary metric which is 1 if the beats are correctly tracked for at least 25% of the piece, and 0 otherwise. *P-score*, introduced by McKinney, Moelants, Davies, and Klapuri (2007), is the proportion of tracked beats which correctly match an annotated beat, normalised by either the number of tracked beats or the number of annotated beats (whichever is greater). Finally, Hainsworth (2003) describes metrics based on the longest continuously tracked subsection of music. All of the above metrics are used to some extent in beat tracking, and the MIREX Audio Beat Tracking task (MIREX, 2017a) uses all of them. In addition, evaluation is also often presented at twice and half the annotated beat length, to handle models which may track a beat at the wrong metrical level.

By comparison, the evaluation of metrical structure detection and alignment is far less sophisticated. Recall that meter detection and alignment is the organisation of the beats of a given musical performance into a sequence of trees at the bar level, in which each node represents a single note value. The structure of each of these trees is directly related to the music's time signature, where the head of each tree splits into a number of nodes equal to the number of beats per bar, and each of these beat nodes splits into a number of nodes equal to the number of sub-beats per beat. Thus, each time signature uniquely describes a single metrical tree structure as defined by the number of beats per bar and sub-beats per beat in that time signature. The most basic evaluation is to simply report the proportion of musical excerpts for which the model guesses the correct metrical structure and phase (such that each tree aligns correctly with a single bar). Another approach is to simply report the proportion of musical excerpts for which the model correctly classifies the meter as duple or triple (De Haas & Volk, 2016). Both of these metrics are simplistic, and fail to take into account some idea of partially correct metrical structure trees.

Temperley (2004) proposes a metric which takes into account the level on the metrical tree at which each note lies in order to capture some idea of partial correctness; however, the fact that it is based on detected notes means that it would not be robust to errors in multi-pitch detection and would violate our principle of disjoint penalties. Two novel metrics have been proposed in this thesis: one for metronomically-aligned music data (and thus not directly applicable) in Section 4.3.2.1, and one in Section 4.4.2.3 for use with live performance data. Since this last metric is based solely on bar, beat, sub beat placement in time, it does not rely on any multi-pitch detection results and thus does not violate our principle of disjoint penalties.

5.3.1.4 Note Value Detection

Note value detection is not a widely researched problem, and is related to a combination of note offset time as well as metrical alignment. Nakamura, Yoshii, and Dixon (2017) describe two metrics for the task. One, error rate, is simply the percentage of notes whose transcribed value is incorrect. The other, scale error, takes into account the magnitude of the error as well (relative to the metrical grid), in log space such that errors from long notes do not dominate the calculation.

However, since note values are reported relative to the underlying meter, and are measured for each detected note, they violate our property of disjoint penalties and we must design a new measure of note value detection accuracy for our metric.

5.3.1.5 Harmonic Analysis

Harmonic analysis involves both key detection, a classification problem of identifying one of twelve tonic notes, each with two possible modes (major or minor—alternate mode detection has not been widely researched); and chord tracking, identifying a sequence of chords and times given an audio recording. The possible chords to identify range from simply identifying the correct root note, to determining major or minor, identifying seventh chords, and even identifying different chord inversions.

The standard key detection evaluation, used both in the mir_eval package (Raffel et al., 2014) and the MIREX Audio Key Detection task (MIREX, 2017d), is to assign a guess a score of 1.0 if it is the correct key, 0.5 if it is a perfect fifth higher than the correct key, 0.3 if it is the relative major or minor of the correct key, 0.2 if it is the parallel major or minor of the correct key, and 0.0 otherwise.

The standard chord tracking evaluation is *chord symbol recall* (CSR)—described by Harte (2010), and used for the MIREX Audio Chord Estimation task (MIREX, 2017b)—which is defined as the proportion of the total duration of the annotated input during which the annotated chord matches the ground truth chord. There can be varying levels of specificity for what exactly constitutes a match, since different sets of possible chords can be used as described above.

Regardless, since neither the described key detection evaluation nor CSR takes into consideration the detected notes or the metrical structure in its calculation, they can each be applied directly to the full AMT task without violating our property of disjoint penalties.

5.3.1.6 Joint Metric

For the joint evaluation of AMT performance, Cogliati, Temperley, and Duan (2016) present a system to transcribe MIDI input into a musical score (thus including errors from typesetting), and evaluate it using five human evaluators. The evaluators were asked to: "1) Rate the pitch notation with regard to the key signature and the spelling of notes. 2) Rate the rhythmic notation with regard to the time signature, bar lines, and rhythmic values. 3) Rate the notation with regard to stems, voicing, and placement of notes on staves," each on a scale of 1 to 10. Notice that the three questions roughly correspond with four of our subsections above: 1) harmonic analysis, Section 5.3.1.5; 2) metrical alignment and note value detection, Sections 5.3.1.3 and 5.3.1.4; and 3) voice separation, Section 5.3.1.2.

Cogliati and Duan (2017) describe an automatic metric for the same task, similar to string edit distance, taking into account the ordering of 12 different aspects of a musical score: barlines, clefs, key signatures, time signatures, notes, note spelling, note durations, stem directions, groupings, rests, rest duration, and staff assignment. While it is a great step towards an automatic evaluation of AMT performance, it violates our principle of disjoint penalties. A single mistake in metrical alignment can manifest itself in the time signature, rest durations, note durations, and even additional notes (tied notes are counted as separate objects in the metric).

It appears that both of the above metrics measure something slightly different from what we want. They measure the readability of a score produced by an AMT system, while we really want a metric which measures the accuracy of the analysis performed by the AMT system, a slightly different task. To our knowledge, no metric exists which measures the accuracy of the analysis performed by a complete AMT system in the way we desire.

5.3.2 New Metric

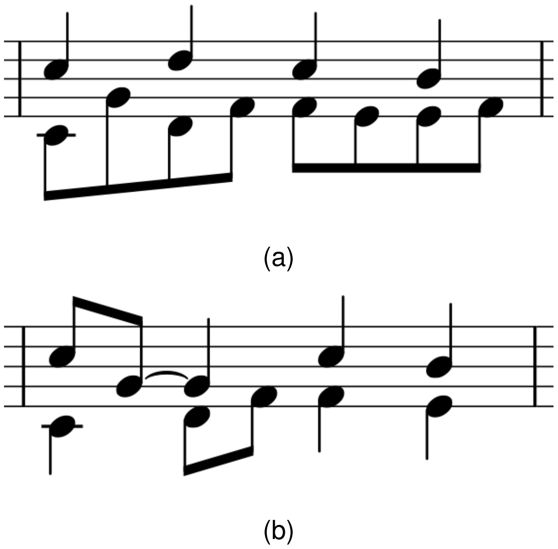
Our proposed metric, $MV2H$, draws from existing metrics where possible, though we take care to ensure that our principle of disjoint penalties is not violated. Essentially, we calculate a single score for each aspect of the transcription, and then combine them all into the final joint metric.

5.3.2.1 Multi-pitch Detection

For multi-pitch detection, we use an F-measure very similar to the one by Bay et al. (2009) described above, counting a detected note as a true positive if its detected pitch (in semitones) is correct and its onset lies within 50 ms of the ground truth onset time. All other detected notes are false positives, and any unmatched ground truth notes are false negatives. Note offset time does not factor into our evaluation; rather, see Section 5.3.2.4 for a discussion on the related problem of note value detection.

5.3.2.2 Voice Separation

For voice separation, we use an F-measure very similar to the one from Section 3.2.3.3, taking care not to violate our disjoint penalties principle. Specifically, we don't want to penalise any model in voice separation for multi-pitch detection errors.



(a)

(b)

Figure 5.3: To illustrate the new voice separation metric, an example transcription of the ground truth bar shown in 5.3a is given in 5.3b. Here, the connection between the last two notes in the lower voice is considered a true positive, even though they are not consecutive in the ground truth.

Recall that the F-measure is calculated as a binary classification problem where for each ordered pair of notes, we must decide if they occur consecutively in the same voice or not. To enforce the disjoint penalties principle, we change this definition slightly. We first remove from both the ground truth voices and the detected voices any notes which have not been matched as a true positive. Then, we perform the same graph-based F-measure calculation with the reduced ground truth and detected voices.

As an illustration of this, see Figure 5.3. In the transcribed music, the last two notes in the lower voice are both matched with a ground truth note (in pitch and onset time), but are not immediately sequential in the ground truth voice. However, because the intervening note was not correctly transcribed, the link between these two notes counts as a true positive. This new F-measure calculation is equivalent to the standard voice separation F-measure when multi-pitch detection is performed perfectly.

5.3.2.3 Metrical Alignment

For our metrical alignment metric, we will use the metrical F-measure proposed in Section 4.4.2.3, where each detected metrical grouping is compared to the ground truth metrical groupings and counted as a true positive if its beginning and end times each lie within 50 ms of a particular ground truth grouping.[2] This metric does not violate our principle of disjoint penalties since it is time-based rather than note-based.

5.3.2.4 Note Value Detection

It is difficult to disentangle note value detection from multi-pitch detection, voice separation, and metrical alignment in order to include it in our evaluation without violating the principle of disjoint penalties. Clearly, note value should only be regarded if the note has been counted as a true positive in the multi-pitch detection evaluation. Less obviously, we also disregard any detected note which is not followed in its hypothesised voice by another detected note corresponding to the next note in the original ground truth voice (not the reduced ground truth voice used for the voice separation F-measure calculation). Additionally, note value depends directly on meter such that any note value accuracy metric must measure note value relative to time rather than relative to the underlying metrical grid.

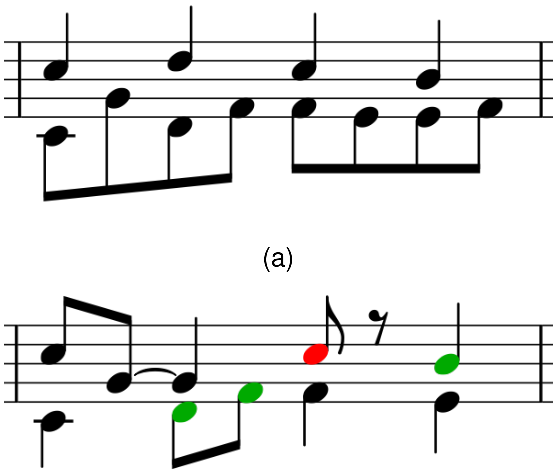
Therefore, we define a note value score which measures only a subset of the detected notes: those which both (1) correspond with a true positive multi-pitch detection; and (2) correspond with a true positive ground truth voice segment as described in the previous paragraph. Each note which matches those two criteria is assigned a score according to the accuracy of its normalised duration (that is, the duration corresponding to its note value rather than its performed duration). Specifically, each note is counted as correct and assigned a score of 1 if its normalised duration is within 100 ms of the normalised duration of the corresponding ground truth note.[3] Otherwise, its score is calculated as in Equation 5.1, where dur_{gt} is the ground truth note's normalised duration and dur_{det} is the detected note's normalised duration. This score is 1 when the durations match exactly and scales linearly on both sides to a score of 0 for a note with 0 duration or a note with twice the ground truth note's duration. The overall note value score is calculated as the arithmetic mean of the scores of those notes which are assigned a score.

$$score = max\left(0, 1 - \frac{|dur_{gt} - dur_{det}|}{dur_{gt}}\right) \tag{5.1}$$

Figure 5.4 illustrates this note value score. In particular, only those notes which

[2] We use a 50 ms threshold (here and elsewhere), rather than the more common 70 ms, because it was shown by (Davies & Böck, 2014) that 50 ms corresponds more exactly with human judgement for beat tracking.

[3] We use 100 ms here to allow for a 50 ms error in both onset and offset time.



(a)

(b)

Figure 5.4: To illustrate the new note value metric, an example transcription of the ground truth bar shown in 5.4a is given in 5.4b. Here, those notes which are assigned a score of 1 are coloured in green, while those notes assigned a score less than 1 are red. Black notes are not considered for the note value detection evaluation.

are coloured are considered for the note value score. Notice that the two C's on the downbeat are not considered due to errors in voice separation. Likewise, the last two notes in the lower voice are also not considered due to voice separation errors, even though they count as a true positives for the voice separation F-measure. The three green notes would be assigned a score of 1, while the red note would be assigned a score of around 0.5 (depending on exact timing), since its value duration is off by exactly half of the ground truth note's value duration. Thus the final note value score would be the average of 1, 1, 1, and 0.5, or about 0.875.

5.3.2.5 Harmonic Analysis

For harmonic analysis, we use the standard key detection and CSR metrics described above, as neither one violates the disjoint penalties principle since they are based on time rather than notes or the metrical alignment. For our purposes, we take our set of possible chords to include a major and minor version for each root note, but not sevenths or inversions.

To combine the two into a single harmonic analysis score, we take the arithmetic mean of the two values, since they are both on the range [0–1]. Models which only perform one of the above tasks may simply use that task's score as their harmonic analysis score.

5.3.2.6 Joint Metric

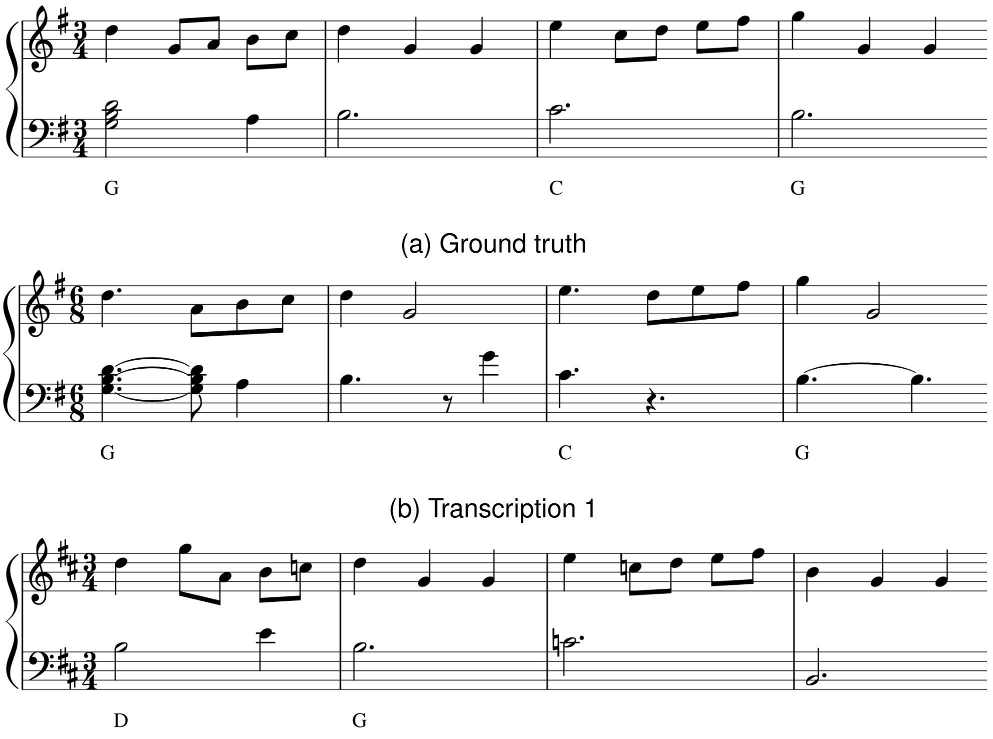
We now have five values to combine into a single number for our final joint metric: the multi-pitch detection F-measure, the voice separation F-measure, the metrical F-measure, the note value detection accuracy score, and the harmonic analysis mean. All of these values are on the range [0–1] such that a value of 1 results from a perfect transcription in that aspect. We consider three different approaches for their combination: harmonic mean, geometric mean, and arithmetic mean.

The harmonic mean is most useful when there is potential for one of the values involved to be significantly larger than the others, and thus dominate the overall result. F-measure, for example, is the harmonic mean between precision and recall, and is used so that models cannot receive a high F-measure by simply tuning their model to have a very high recall or precision; rather, both recall and precision must be relatively high in order for their harmonic mean to also be high. This is not relevant in our case as there is no way for a model to tune itself towards one very high score at the expense of the others as is the case with some binary classification problems.

The geometric mean is most useful when the values involved are on different scales. Then, a given percent change in one of the values will result in the same change in mean as the same percent change to another of the values. This property is not necessary for us because all of our values lie on the same range.

Arithmetic mean, on the other hand, is a simple calculation that weights each of the values involved equally. This property is desirable for us because, for a complete transcription, all five aspects of an analysis must be correct. Furthermore, due to our property of disjoint penalties, we have kept the five values involved disjoint, and a model must fairly perform well on all aspects in order for its overall score to be high.

Therefore, for the final joint metric, $MV2H$ (from **M**ulti-pitch detection, **V**oice separation, **M**etrical alignment, note **V**alue detection, and **H**armonic analysis), we take the arithmetic mean of the five previously calculated values. We also want the metric to be usable no matter what subset of analyses is performed, for example, for models which run on MIDI input and therefore do not perform multi-pitch detection. In these cases, we advise using our metric and simply taking the arithmetic mean of only those scores which correspond with analyses performed.



(a) Ground truth

(b) Transcription 1

(c) Transcription 2

Figure 5.5: Two different example transcriptions of the first four bars of Bach's Minuet in G. The ground truth transcription is given in 5.5a, and the example transcriptions are shown below. The chord progression is given beneath each staff.

5.3.3 Examples

To illustrate the effectiveness and appropriateness of our metric, we present in Figure 5.5 two example transcriptions of the first four bars of Bach's Minuet in G, each exhibiting different errors. Figure 5.5a shows the ground truth transcription (where the chord progression is shown beneath the staff), and the example transcriptions are shown below it.

Figure 5.5b shows an example transcription which is good in general, with just a few mistakes, mostly related to the metrical alignment. First, for the multi-pitch detection F-measure, we can see that the transcription has 20 true positives, 3 false negatives (a G on the second beat in the first bar, a C on the second beat of the third bar, and the final G in the fourth bar), and 0 false positives, resulting in an F-measure of 0.93. For voice separation, this transcription is generally good, making a single bad assignment in the second bar, resulting in 3 false positives (the connections to and from the incorrect assignment, as well as the incorrect connection in the treble clef), 3 false negatives (the correct connections to and from the misclassified note, as well as the correct connection in the bass clef), and a voice separation F-measure of 0.83. Notice that the missed G in the upper voice in the treble clef of the first bar does not result in a penalty for voice assignment due to our principle of disjoint penalties. For metrical alignment, we can see that this transcription is notated in $\frac{6}{8}$ time, correctly grouping all sub beats (eighth notes) and bars, yielding 28 true positives, but incorrectly grouping three sub beats together into dotted quarter note beats, yielding 8 false positives and 12 false negatives. This results in a metrical F-measure of 0.74. For note value detection, 14 notes are counted: all of the bass clef notes and all of the eighth notes in the first bar, only the high D in the second bar, the low C and all of the eighth notes in the third bar, and the high G and the low B in the fourth bar. Notice that the initial high D isn't counted because the next note in its voice has not been detected. Similarly, neither the G on the second beat of the second bar nor any of the bass clef notes in the second bar are counted due to voice separation errors. Of the 14 notes, 13 of them are assigned the correct note value (even the first bass chord, since its incorrect typesetting and the ties are related to the incorrect metrical alignment—the note value still ends at the correct point in time). One note (the C in the bass clef on the downbeat of the third bar) is assigned a value score of 0.5 (since its value duration is half of the correct value duration). This results in a note value detection score of 0.96. The harmonic analysis in this transcription is entirely correct, resulting in a harmonic score of 1.0. Thus, the $MV2H$ of the first transcription is 0.89. This makes sense because the transcription is quite good in general, but a few mistakes are made, the most glaring of which is the metrical alignment (hence that is its lowest individual score).

Figure 5.5c shows another example transcription which is again good in general, this time with a few more errors in multi-pitch detection, as well as a poor harmonic analysis. For multi-pitch detection, it contains 17 true positives, 4 false positives, and 6 false negatives, resulting in an F-measure of 0.77. This number is 0.16 lower than that the previous transcription's corresponding F-measure, and this makes sense intuitively: the first transcription does seem to have resulted from a more accurate multi-pitch detection than the second. For voice separation, this second transcription contains no errors. Some erroneous notes are placed into one voice or the other, but all of the correctly detected notes are also correctly separated into voices, resulting in a perfect voice separation F-measure of 1.0. Likewise the metrical alignment is performed perfectly, resulting in a metrical F-measure of 1.0. For note value detection, we look at all of the true positive note detections except (1) the initial D on the downbeat of the

Transcription	1	2
Multi-pitch	**0.93**	0.77
Voice	0.83	**1.0**
Meter	0.74	**1.0**
Value	0.96	**1.0**
Harmonic	**1.0**	0.5
MV2H	**0.89**	0.85

Table 5.2: The resulting scores from each of the example transcriptions from Figure 5.5.

first bar, (2) the B in the bass clef of the first bar, (3) the C in the bass clef of the third bar, and (4) the high F at the end of the third bar. (All of these exceptions are due to missed note detections of the following note in each voice.) All of the remaining notes have been assigned the correct value, resulting in a note value detection score of 1.0. For the harmonic analysis, the model has incorrectly transcribed the excerpt in D major, resulting in a key score of 0.5. Likewise, the model has incorrectly labelled the chord progression as D-G-G-G, rather than G-G-C-G. Thus, it has transcribed the correct chord for half of the transcription, resulting in a CSR of 0.5, and a harmonic score of 0.5. The *MV2H* of the second transcription is therefore 0.85: slightly worse than the first transcription, but still good.

The scores of both transcriptions are summarised in Table 5.2, and intuitively, they make sense. Both seem good overall, though they both contain errors. The first transcription has an incorrectly notated meter (although its bars and sub beats still align correctly) and a few other smaller mistakes related to multi-pitch detection, voice separation, and note value detection. The second transcription, on the other hand, correctly aligns the meter, and makes its only errors in its harmonic analysis (which is quite poor), and in multi-pitch detection (it is worse than the first model in this regard). Given these examples, for applications which need a good all-around transcription, we would recommend the system which produced the first transcription. However, applications which emphasise metrical structure detection or voice separation should consider using the system which produced the second transcription instead.

We also present in Table 5.3 the results from the different joint versions of our model. This table is essentially just a reproduction of Table 5.1 along with the added joint metric. The joint metric in this case is simply the average of the two numbers

Model	Voice	Meter	Joint
Independent	94.34	56.51	75.43
Sequential	**94.56**	**57.33**	**75.95**
Joint	85.78	27.77	56.78

Table 5.3: The results of our joint model on both voice separation and metrical alignment, as well as our new joint evaluation, compared against those of our two baselines: Independent, which runs each individual model entirely independently; and Sequential, which runs both individual models in sequence rather than simultaneously.

reported individually, since the combination of voice separation and metrical detection does not violate the principle of disjoint penalties.

5.4 Conclusion

In this chapter, we have evaluated running the two components of our music language model—voice separation (see Chapter 3) and metrical alignment (see Chapter 4)—jointly rather than independently as we did in the previous chapters.

We have shown that running the models sequentially leads to a small but noticeable improvement in both voice separation and metrical alignment performance. This suggests, as we have hypothesised earlier, that the components are able to inform and improve each other when integrated properly. Furthermore, it demonstrates that our eventual goal of a full music language model to be run jointly with a multi-pitch detection model is a worthwhile one, and has the potential to add real value to the field of AMT.

As future work moves towards this goal of a complete AMT system, an automatic, standardised, quantitative metric for the task will become a necessity. To that end, we have proposed a joint metric, $MV2H$, which measures multi-pitch detection, voice separation, metrical alignment, note value detection, and harmonic analysis and summarises them in a single number. Our metric is based on the property of disjoint penalties: that a model should not be penalised twice for errors which come from a single mistake or misinterpretation. While our metric may not be the final standardised metric used for the task, we believe that it should become part of the discussion, and the that the principles that guided us through its creation should continue to be addressed by any future proposed metrics. In future work, we will investigate whether a linear combination of the five values involved, perhaps weighting some more strongly than others, aligns more exactly with human judgements than the current arithmetic mean.

Future work on our joint model will concentrate on its simplification and integration. Specifically, the complexity of the model has proven to be a barrier to its joint inference, requiring a beam size too large to be computationally feasible in a reasonable amount of time. Smarter and more aggressive pruning of potential hypotheses at each step, for example by calculating upper bound probability estimates for partial metrical trees and using heuristics as in A* search, should bring the required beam size down to a reasonable level, and future research will have that goal in mind.

Chapter 6

Conclusion

This thesis has argued for a new approach to language modelling for music based on natural language processing (NLP) techniques, and the applicability of such a language model design to the improvement of automatic music transcription (AMT) performance. We began with a look at the performance of state-of-the-art AMT systems, in particular their lack of significant progress in recent years, and argued that language modelling could be the key to future progress. It has been shown before that music and language exhibit similarities, and it is therefore natural to design a music language model based on NLP techniques. This thesis has presented such a language model for music analysis. While our proposed model is at times inspired by human cognition (which is natural since humans currently outperform computers on many transcription tasks), we do not present them as cognitive models of the human interpretation of music. Our goal in this thesis is instead for them to be used to improve AMT performance.

Chapter 2 offered an overview of existing work on multi-pitch detection, the first step in the process of transcribing audio data, which involves converting an input audio file into a time frequency format such as a piano roll or a MIDI file. While the subject of multi-pitch detection is not directly relevant to this thesis, a strong knowledge of the field does help to give context to the remaining chapters. In particular, it directs us towards the creation of certain principles which should guide the creation of any music language model. In particular, we have learned that a music language model should (1) be probabilistic; (2) not use any information besides that which is output directly by a multi-pitch detection system (note pitch and timing information); (3) have the ability to run directly on live performance data without any preprocessing or data cleanup; and (4) be incremental, working from the beginning of the song to the end. These four principles have guided us through the remaining chapters during the creation of our music language model.

Chapter 3 presented a model for voice separation and assignment, the process of taking an input polyphonic stream of notes (which may contain many simultaneous notes) and separating those notes into labelled monophonic streams of notes (which contain only one note at a time). Voice separation is often used as a preprocessing step for further musical analysis. We showed that our model achieves state-of-the-art voice separation performance on datasets of both metronomic and live performance MIDI, despite requiring no a priori information or alignment for the input piece (unlike many of the models we compared against). Thus, our approach is both the most accurate, but also able to be run directly on any input stream of notes, including live performance. We also extended this model, integrating it with a multi-pitch detection model into a system for transcription and voice assignment of four-part a cappella recordings. Notably, our combined system sees a significant improvement over a version of the system without the voice separation component, and achieves state-of-the-art results for both multi-pitch detection and voice assignment. This illustrates that voice assignment is not only useful as a preprocessing step for more complicated music analysis tasks. Rather, it is able to provide real value as a music language model itself.

Chapter 4 proposed a model for meter detection and alignment, the process of aligning an input musical performance with repeating metrical tree structures with one tree per bar. Essentially, our model aligns the notes of the performance with a time signature and its pattern of bars, beats, and sub beats. The model consists of two components. The first, a lexicalised probabilistic context-free grammar (LPCFG), can be used to perform metrical alignment directly on metronomic MIDI data which is aligned with some pulse (for example, 32nd notes) by creating rhythmic parse trees. The basic idea of the grammar is to detect strong and weak beats and sub beats according to rhythmic stress of the input notes, and then align that stress pattern with a particular repeating metrical tree structure. A standard PCFG makes a strong independence assumption that the rhythmic stress which occurs in one position in the tree does not affect the stress at any other position in the tree. Lexicalisation, a technique used in natural language parsing, breaks this assumption, allowing the grammar to draw from a wider context when detecting rhythmic stress. We showed that the LPCFG significantly outperforms a standard PCFG, thus demonstrating that musical rhythms display long-distance dependencies similar to those found in natural language. We combined this LPCFG with an HMM to create an incremental, probabilistic model for metrical alignment of live performance data which requires no information a priori besides a list of note pitches and timings. We also proposed new metrics for metrical alignment of both metronomic and live performance MIDI which take into account some idea of partial correctness. Using the new metrics, we showed that our combined model achieves state-of-the-art metrical alignment performance on both types of data. This again shows that our approach to metrical alignment using NLP-inspired parsing methods for language has merit, and that music exhibits many of the same patterns and structures as natural language.

Finally, Chapter 5 investigated the joint modelling and analysis of music. First, we integrated our voice separation model from Chapter 3 with our metrical alignment model from Chapter 4 into a single joint model to perform both tasks. We showed that running the two models sequentially and taking the most probable combined result results in improved performance compared with running each model independently and taking the most probable result at each step. This result demonstrates that each aspect of the musical analysis is able to inform and improve the other, suggesting that a joint analysis of music (even incorporating multi-pitch detection where possible) should be preferred to independent analyses of each aspect. To that end, we also proposed $MV2H$, a new automatic, quantitative metric for a complete transcription of polyphonic music consisting of multi-pitch detection, voice separation, metrical alignment, note value detection, and harmonic analysis. Our new metric is based on the property of double jeopardy: that a model should not be penalised twice for two transcription errors which both stem from the same mistake. We believe that double jeopardy is an important property and that any metric which becomes standard for the task should take it into consideration.

The contribution of this thesis is to show that a music language model based on NLP techniques is able to model music well enough to offer improvement to AMT performance for multi-pitch detection, voice separation, and metrical alignment. This outcome suggests a connection between the structures of music and language. The models used here were not only able to achieve state-of-the-art performance on their own tasks, but they were also able to improve both multi-pitch detection accuracy (a major component of AMT), and even improve each other's performance when run jointly. The complexity of the models does not allow the full model to be run jointly along with a multi-pitch detection program, but its accuracy has shown that with a more sophisticated integration procedure, such a joint model should improve overall transcription performance. To that end, this thesis has also proposed a new automatic, quantitative metric for the task of a full transcription and analysis of music, a necessary aspect of a field which can be expected to be the subject of much research in the years to come.

The models developed as a part of this work take as input an unlabelled, metrically unaligned stream of note pitches, onsets, and offsets, and give the stream structure. They separate the input into more manageable monophonic voices, and align a metrical structure to the notes. Such analysis is applicable to the field of music information retrieval both directly, for transcribing MIDI performance, and indirectly, as features for downstream applications such as query-by-tapping, query-by-humming, or even genre detection and music recommendation. Furthermore, we have shown that a music language model such as our voice separation model can aid in multi-pitch detection from audio, and it is therefore reasonable to expect that our combined model could be an even better music language model for multi-pitch detection—and even become the starting point for the creation of a complete music transcription system: from audio recording to musical score.

References

Bay, M., Ehmann, A. F., Beauchamp, J. W., Smaragdis, P., & Downie, J. S. (2012). Second fiddle is important too: Pitch tracking individual voices in polyphonic music. In *International Society of Music Information Retrieval Conference (ISMIR)* (pp. 319–324).

Bay, M., Ehmann, A. F., & Downie, J. S. (2009). Evaluation of multiple-F0 estimation and tracking systems. In *ISMIR* (pp. 315–320).

Bello, J. P., Daudet, L., & Sandler, M. B. (2006, November). Automatic piano transcription using frequency and time-domain information. *IEEE Transactions on Audio, Speech and Language Processing*, *14*(6), 2242–2251. doi: 10.1109/TASL.2006.872609

Benetos, E., Badeau, R., Weyde, T., & Richard, G. (2014). Template adaptation for improving automatic music transcription. In *ISMIR* (pp. 175–180).

Benetos, E., & Dixon, S. (2012, December). A shift-invariant latent variable model for automatic music transcription. *Computer Music Journal*, *36*(4), 81–94. doi: 10.1162/COMJ_a_00146

Benetos, E., & Dixon, S. (2013). Multiple-instrument polyphonic music transcription using a temporally constrained shift-invariant model. *The Journal of the Acoustical Society of America*, *133*(3), 1727-1741.

Benetos, E., Dixon, S., Giannoulis, D., Kirchhoff, H., & Klapuri, A. (2013, July). Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, *41*(3), 407–434. doi: 10.1007/s10844-013-0258-3

Benetos, E., & Holzapfel, A. (2015). Automatic transcription of turkish microtonal music. *The Journal of the Acoustical Society of America*, *138*(4), 2118–2130.

Benetos, E., & Weyde, T. (2015). An efficient temporally-constrained probabilistic model for multiple-instrument music transcription. In *ISMIR* (pp. 701–707).

Berg-Kirkpatrick, T., Andreas, J., & Klein, D. (2014). Unsupervised transcription of piano music. In *Advances in Neural Information Processing Systems* (pp.

1538–1546).

Bernstein, L. (1976). *The unanswered question: Six talks at harvard* (Vol. 33). Harvard University Press.

Birmingham, W., Dannenberg, R., & Pardo, B. (2006). Query by humming with the vocalsearch system. *Communications of the ACM*, *49*(8), 49–52.

Bittner, R. M., McFee, B., Salamon, J., Li, P., & Bello, J. P. (2017). Deep salience representations for F0 estimation in polyphonic music. In *ISMIR* (pp. 63–70).

Böck, S., & Schedl, M. (2012, March). Polyphonic piano note transcription with recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 121–124). IEEE. doi: 10.1109/ICASSP.2012.6287832

Bogdanov, D., Wack, N., Gómez, E., Gulati, S., Herrera, P., Mayor, O., … Serra, X. (2013). Essentia: An audio analysis library for music information retrieval. In *ISMIR* (pp. 493–498).

Bohak, C., & Marolt, M. (2016). Transcription of polyphonic vocal music with a repetitive melodic structure. *Journal of the Audio Engineering Society*, *64*(9), 664–672.

Boulanger-Lewandowski, N., Bengio, Y., & Vincent, P. (2013, May). High-dimensional sequence transduction. In *ICASSP* (pp. 3178–3182). IEEE. doi: 10.1109/ICASSP.2013.6638244

Brown, J. C. (1991, January). Calculation of a constant Q spectral transform. *J. Acoustical Society of America*, *89*(1), 425-434.

Brown, J. C. (1993). Determination of the meter of musical scores by autocorrelation. *The Journal of the Acoustical Society of America*, *94*(4), 1953–1957. doi: 10.1121/1.407518

Bruderer, M., McKinney, M., & Kohlrausch, A. (2012). Perceptual evaluation of musicological cues for automatic song segmentation. *Psychomusicology: Music, Mind and Brain*, *22*(1), 3.

Cambouropoulos, E. (2008, September). Voice and stream: Perceptual and computational modeling of voice separation. *Music Perception*, *26*(1), 75–94. doi: 10.1525/mp.2008.26.1.75

Cemgil, A. T., Kappen, B., Desain, P., & Honing, H. (2000, December). On tempo tracking: Tempogram representation and kalman filtering. *Journal of New Music Research*, *29*(4), 259–273. doi: 10.1080/09298210008565462

Chew, E., & Wu, X. (2004). Separating voices in polyphonic music: A contig mapping approach. In *Computer music modeling and retrieval* (pp. 1–20). doi: 10.1007/b105507

Cogliati, A., & Duan, Z. (2017). A metric for music notation transcription accuracy. In *ISMIR* (pp. 407–413).

Cogliati, A., Temperley, D., & Duan, Z. (2016). Transcribing human piano performances into music notation. In *ISMIR* (pp. 758–764).

Davies, M. E. P., & Böck, S. (2014). Evaluating the evaluation measures for beat tracking. In *ISMIR* (pp. 637–642).

Davies, M. E. P., Degara, N., & Plumbley, M. D. (2009). Evaluation methods for musical audio beat tracking algorithms. *Queen Mary University of London, Centre for Digital Music, Technical Report C4DM-TR-09-06*.

De Haas, W. B., & Volk, A. (2016). Meter detection in symbolic music using inner metric analysis. In *ISMIR* (pp. 441–447).

de A. Scatolini, C., Richard, G., & Fuentes, B. (2015, April). Multipitch estimation using a PLCA-based model: Impact of partial user annotation. In *ICASSP* (pp. 186–190).

de León, P., & Inesta, J. (2007). Pattern recognition approach for music style identification using shallow statistical descriptors. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *37*(2), 248–257.

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

de Valk, R. (2015). *Structuring lute tablature and MIDI data: Machine learning models for voice separation in symbolic music representations* (Unpublished doctoral dissertation). City University of London.

Dixon, S. (2001, March). Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, *30*(1), 39–58. doi: 10.1076/jnmr.30.1.39.7119

Dixon, S. (2007, March). Evaluation of the audio beat tracking system beatroot. *Journal of New Music Research*, *36*(1), 39–50. doi: 10.1080/09298210701653310

Duan, Z., Han, J., & Pardo, B. (2014, January). Multi-pitch streaming of harmonic sound mixtures. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *22*(1), 138–150.

Duane, B., & Pardo, B. (2009). Streaming from MIDI using constraint satisfaction optimization and sequence alignment. In *International Computer Music Conference (ICMC)* (pp. 1–8).

Eck, D., & Casagrande, N. (2005). Finding meter in music using an autocorrelation phase matrix and shannon entropy. In *ISMIR* (pp. 504–509).

Elowsson, A., & Friberg, A. (2014). Polyphonic transcription with deep layered learning. In *MIREX.* Retrieved from `http://www.music-ir.org/mirex/abstracts/2014/EF1.pdf`

Fouloulis, T., Pikrakis, A., & Cambouropoulos, E. (2013). Traditional asymmetric rhythms: A refined model of meter induction based on asymmetric meter templates. In *Proceedings of the third international workshop on folk music analysis* (pp. 28–32).

Fuentes, B., Badeau, R., & Richard, G. (2012, August). Blind harmonic adaptive decomposition applied to supervised source separation. In *European Signal Processing Conference (EUSIPCO)* (pp. 2654–2658).

Fuentes, B., Badeau, R., & Richard, G. (2014, September). Controlling the convergence rate to help parameter estimation in a PLCA-based model. In *(EUSIPCO)* (p. 626-630).

Giannoulis, D., Benetos, E., Klapuri, A., & Plumbley, M. D. (2014). Improving instrument recognition in polyphonic music through system integration. In *ICASSP* (pp. 5222–5226).

Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 237–264.

Goto, M. (2004). A real-time music-scene-description system: Predominant-F0 estimation for detecting melody and bass lines in real-world audio signals. *Speech Communication*, *43*(4), 311–329.

Goto, M., Hashiguchi, H., Nishimura, T., & Oka, R. (2004). RWC music database: Music genre database and musical instrument sound database. In *ISMIR* (pp. 229–230).

Goto, M., & Muraoka, Y. (1997). Issues in evaluating beat tracking systems. In *Workshop on issues in AI and music* (pp. 9–16).

Gould, E. (2011). *Behind bars : the definitive guide to music notation.* Faber Music.

Granroth-Wilding, M., & Steedman, M. (2014, June). A robust parser-interpreter for jazz chord sequences. *Journal of New Music Research*, *43*(4), 355–374.

Gray, P., & Bunescu, R. (2016). A neural greedy model for voice separation in symbolic music. In *ISMIR* (pp. 782–788).

Grindlay, G., & Ellis, D. P. W. (2011, October). Transcribing multi-instrument polyphonic music with hierarchical eigeninstruments. *IEEE Journal on Selected*

Topics in Signal Processing, *5*(6), 1159-1169.

Guiomard-Kagan, N., Giraud, M., Groult, R., & Levé, F. (2016). Improving voice separation by better connecting contigs. In *ISMIR* (pp. 164–170).

Hainsworth, S. W. (2003). *Techniques for the automated analysis of musical audio* (Unpublished doctoral dissertation). University of Cambridge.

Hanna, P., & Robine, M. (2009). Query by tapping system based on alignment algorithm. In *ICASSP* (pp. 1881–1884).

Harte, C. (2010). *Towards automatic extraction of harmony information from music signals* (Unpublished doctoral dissertation). Queen Mary University of London.

Hashida, M., Matsui, T., & Katayose, H. (2008). A new music database describing deviation information of performance expressions. *ISMIR*, 489–494.

Hsu, J.-L., Liu, C.-C., & Chen, A. L. P. (2001). Discovering nontrivial repeating patterns in music data. *IEEE Transactions on Multimedia*, *3*(3), 311–325. doi: 10.1109/6046.944475

Huron, D. (2001, September). Tone and voice: A derivation of the rules of voice-leading from perceptual principles. *Music Perception*, *19*(1), 1–64.

Ishigaki, A., Matsubara, M., & Saito, H. (2011). Prioritized contig combining to segregate voices in polyphonic music. In *Sound and Music Computing Conference (SMC)* (pp. 58–64).

Jordanous, A. (2008, August). Voice separation in polyphonic music: A data-driven approach. In *ICMC*.

Jurafsky, D., & Martin, J. H. (2000). Speech and language processing. *International Edition*.

Kameoka, H., Nakano, M., Ochiai, K., Imoto, Y., Kashino, K., & Sagayama, S. (2012). Constrained and regularized variants of non-negative matrix factorization incorporating music-specific constraints. In *ICASSP* (pp. 5365–5368).

Kameoka, H., Nishimoto, T., & Sagayama, S. (2007, March). A multipitch analyzer based on harmonic temporal structured clustering. *IEEE Transactions on Audio, Speech and Language Processing*, *15*(3), 982–994.

Karydis, I., Nanopoulos, A., Papadopoulos, A., Cambouropoulos, E., & Manolopoulos, Y. (2007). Horizontal and vertical integration/segregation in auditory streaming: a voice separation algorithm for symbolic musical data. In *SMC* (pp. 299–306).

Kelz, R., Dorfer, M., Korzeniowski, F., Böck, S., Arzt, A., & Widmer, G. (2016). On the potential of simple framewise approaches to piano transcription. In *ISMIR* (pp. 475–481).

Kilian, J. (2004). *Inferring score level musical information from low-level musical data* (Unpublished doctoral dissertation). TU Darmstadt.

Kilian, J., & Hoos, H. (2002). Voice separation—a local optimization approach. In *ISMIR*.

Kirchhoff, H., Dixon, S., & Klapuri, A. (2013). Missing template estimation for user-assisted music transcription. In *ICASSP* (pp. 26–30).

Kirlin, P., & Utgoff, P. (2005). VOISE: Learning to segregate voices in explicit and implicit polyphony. In *ISMIR* (pp. 552–557).

Klapuri, A. (2005). A perceptually motivated multiple-F0 estimation method. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (pp. 291–294). IEEE. doi: 10.1109/ASPAA.2005.1540227

Klapuri, A. (2006). Multiple fundamental frequency estimation by summing harmonic amplitudes. In *ISMIR* (pp. 216–221).

Lee, C. (1991, May). The perception of metrical structure: Experimental evidence and a new model. In P. Howell, R. West, & I. Cross (Eds.), *Representing musical structure* (pp. 59–128). Academic Press. doi: 10.1121/1.2024471

Lee, D. D., & Seung, H. S. (1999, October). Learning the parts of objects by non-negative matrix factorization. *Nature*, *401*, 788–791.

Lerdahl, F., & Jackendoff, R. S. (1985). *A generative theory of tonal music*. MIT press.

Longuet-Higgins, H. C. (1972). *The language of music*. (Unpublished manuscript)

Longuet-Higgins, H. C., & Lee, C. S. (1982). The perception of musical rhythms. *Perception*, *11*(2), 115–128. doi: 10.1068/p110115

Longuet-Higgins, H. C., & Steedman, M. (1971). On interpreting Bach. *Machine Intelligence*, *6*, 221–241.

Madsen, S. T., & Widmer, G. (2006). Separating voices in MIDI. *ISMIR*, 57–60.

Mauch, M., & Dixon, S. (2014). PYIN: A fundamental frequency estimator using probabilistic threshold distributions. In *ICASSP* (pp. 659–663).

McKinney, M. F., Moelants, D., Davies, M. E. P., & Klapuri, A. (2007, March). Evaluation of audio beat tracking and music tempo extraction algorithms. *Journal of New Music Research*, *36*(1), 1–16. doi: 10.1080/09298210701653252

McLeod, A., Schramm, R., Steedman, M., & Benetos, E. (2017). Automatic Transcription of Polyphonic Vocal Music. *Applied Sciences*, *7*(12).

McLeod, A., & Steedman, M. (2016, January). HMM-based voice separation of MIDI performance. *Journal of New Music Research*, *45*(1), 17–26.

McLeod, A., & Steedman, M. (2017). Meter detection in symbolic music using a lexicalized PCFG. In *SMC* (pp. 373–379).

McLeod, A., & Steedman, M. (2018a). Evaluating automatic polyphonic music transcription. In *ISMIR*.

McLeod, A., & Steedman, M. (2018b). Meter detection and alignment of MIDI performance. In *ISMIR*.

Meudic, B. (2002). Automatic meter extraction from MIDI files. In *Journées d'informatique musicale.*

Meyer, L. B. (1956). *Emotion and meaning in music.* University of Chicago Press.

MIREX. (2017a). *Audio beat tracking.* `http://www.music-ir.org/mirex/wiki/2017:Audio\_Beat\_Tracking`. (Accessed: 2017-07-18)

MIREX. (2017b). *Audio chord estimation.* `http://www.music-ir.org/mirex/wiki/2017:Audio\_Chord\_Estimation`. (Accessed: 2017-07-18)

MIREX. (2017c). *Audio downbeat estimation.* `http://www.music-ir.org/mirex/wiki/2017:Audio\_Downbeat\_Estimation`. (Accessed: 2017-07-18)

MIREX. (2017d). *Audio key detection.* `http://www.music-ir.org/mirex/wiki/2017:Audio\_Key\_Detection`. (Accessed: 2017-07-18)

MIREX. (2017e). *Multiple fundamental frequency estimation & tracking.* `http://www.music-ir.org/mirex/wiki/2017:Multiple\_Fundamental\_Frequency\_Estimation\_\%26\_Tracking`. (Accessed: 2017-07-18)

Mysore, G. J., & Smaragdis, P. (2009). Relative pitch estimation of multiple instruments. In *ICASSP* (pp. 313–316).

Nakamura, E., Yoshii, K., & Dixon, S. (2017, September). Note value recognition for piano transcription using markov random fields. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *25*(9), 1846–1858.

Nakamura, E., Yoshii, K., & Sagayama, S. (2016). Rhythm transcription of polyphonic MIDI performances based on a merged-output HMM for multiple voices. In *SMC* (pp. 338–343).

O'Hanlon, K., Nagano, H., Keriven, N., & Plumbley, M. D. (2016, March). Nonnegative group sparsity with subspace note modelling for polyphonic transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *24*(3), 530-542. doi: 10.1109/TASLP.2016.2515514

Papadopoulos, H., & Peeters, G. (2011, January). Joint estimation of chords and downbeats from an audio signal. *IEEE Transactions on Audio, Speech, and Language Processing*, *19*(1), 138–152. doi: 10.1109/TASL.2010.2045236

Pertusa, A., & Iñesta, J. M. (2012). Efficient methods for joint estimation of multiple fundamental frequencies in music signals. *EURASIP Journal on Advances in Signal Processing*.

Peters, G., Cukierman, D., Anthony, C., & Schwartz, M. (2006). Online music search by tapping. In *Ambient intelligence in everyday life* (pp. 178–197). Springer.

Poliner, G. E., & Ellis, D. P. W. (2007). A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, *2007*(1), 154–162. doi: 10.1155/2007/48317

Raffel, C., Mcfee, B., Humphrey, E. J., Salamon, J., Nieto, O., Liang, D., ... Humphrey, E. J. (2014). mir_eval: A transparent implementation of common MIR metrics. In *ISMIR*.

Rohrmeier, M. (2011, March). Towards a generative syntax of tonal harmony. *Journal of Mathematics and Music*, *5*(1), 35–53.

Ryynanen, M. P., & Klapuri, A. (2005, October). Polyphonic music transcription using note event modeling. In *Ieee workshop on applications of signal processing to audio and acoustics, 2005.* (p. 319-322). doi: 10.1109/ASPAA.2005.1540233

Ryynanen, M. P., & Klapuri, A. (2008). Query by humming of MIDI and audio using locality sensitive hashing. In *ICASSP* (pp. 2249–2252).

Salamon, J., & Gomez, E. (2012, August). Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(6), 1759–1770.

Schörkhuber, C., Klapuri, A., Holighaus, N., & Dörfler, M. (2014, January). A Matlab toolbox for efficient perfect reconstruction time-frequency transforms with log-frequency resolution. In *AES International Conference on Semantic Audio.*

Schramm, R., & Benetos, E. (2017, June). Automatic transcription of a cappella recordings from multiple singers. In *AES International Conference on Semantic Audio.*

Schramm, R., McLeod, A., Steedman, M., & Benetos, E. (2017). Multi-pitch detection and voice assignment for a cappella recordings of multiple singers. In *ISMIR* (pp. 552–559).

Shashanka, M., Raj, B., & Smaragdis, P. (2008). Probabilistic latent variable models as nonnegative factorizations. *Computational Intelligence and Neuroscience*. (Article ID 947438)

Sigtia, S., Benetos, E., & Dixon, S. (2016, May). An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *24*(5), 927–939. doi: 10.1109/TASLP.2016 .2533858

Spiro, N. (2002). Combining grammar-based and memory-based models of perception of time signature and phase. In *Music and artificial intelligence* (pp. 183–194). Springer Berlin Heidelberg. doi: 10.1007/3-540-45722-4_17

Steedman, M. (1977, January). The perception of musical rhythm and metre. *Perception*, *6*(5), 555–69.

Steedman, M. (1996). The blues and the abstract truth: Music and mental models. *Mental models in cognitive science*, 305–318.

Takeda, H., Nishimoto, T., & Sagayama, S. (2004). Rhythm and tempo recognition of music performance from a probabilistic approach. In *ISMIR*.

Takeda, H., Nishimoto, T., & Sagayama, S. (2007). Rhythm and tempo analysis toward automatic music transcription. In *ICASSP* (pp. 1317–1320). IEEE.

Temperley, D. (2004, September). An evaluation system for metrical models. *Computer Music Journal*, *28*(3), 28–44. doi: 10.1162/0148926041790621

Temperley, D. (2007). *Music and probability*. The MIT Press.

Temperley, D. (2008, February). A probabilistic model of melody perception. *Cognitive Science*, *32*(2), 418–444.

Temperley, D. (2009, March). A unified probabilistic model for polyphonic music analysis. *Journal of New Music Research*, *38*(1), 3–18. doi: 10.1080/ 09298210902928495

Thickstun, J., Harchaoui, Z., Foster, D., & Kakade, S. M. (2017). MIREX 2017: Frequency Domain Convolutions for Multiple F0 Estimation. In *MIREX*. Retrieved from `http://www.music-ir.org/mirex/abstracts/2017/THK1.pdf`

Thickstun, J., Harchaoui, Z., & Kakade, S. (2017). Learning features of music from scratch. In *International Conference on Learning Representations*.

Toiviainen, P., & Eerola, T. (2006). Autocorrelation in meter induction: The role of accent structure. *The Journal of the Acoustical Society of America*, *119*(2), 1164.

Tymoczko, D. (2008, March). Scale theory, serial theory and voice leading. *Music Analysis*, *27*(1), 1–49.

Valero-Mas, J. J., Benetos, E., & Iñesta, J. M. (2016). Classification-based note tracking for automatic music transcription. In *Proceedings of the 9th machine learn-*

ing and music workshop.

van der Weij, B. (2012). *Subdivision-based parsing of expressively performed rhythms* (Unpublished master's thesis). University of Edinburgh.

Vincent, E., Bertin, N., & Badeau, R. (2010, March). Adaptive harmonic spectral decomposition for multiple pitch estimation. *IEEE Transactions on Audio, Speech, and Language Processing*, *18*(3), 528–537. doi: 10.1109/TASL.2009.2034186

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

Volk, A. (2008, December). The study of syncopation using inner metric analysis: Linking theoretical and experimental analysis of metre in music. *Journal of New Music Research*, *37*(4), 259–273.

Volk, A., & de Haas, W. B. (2013). A corpus-based study on ragtime syncopation. *ISMIR*, 163–168.

Weninger, F., Kirst, C., Schuller, B., & Bungartz, H.-J. (2013, May). A discriminative approach to polyphonic piano note transcription using supervised non-negative matrix factorization. In *ICASSP* (pp. 6–10). IEEE. doi: 10.1109/ICASSP.2013.6637598

Whiteley, N., Cemgil, A. T., & Godsill, S. (2006). Bayesian modelling of temporal structure in musical audio. In *ISMIR* (pp. 29–34).

Yeh, C. (2008). *Multiple fundamental frequency estimation of polyphonic recordings* (Unpublished doctoral dissertation). Université Lille 1.

Young, S. (1996, September). A review of large-vocabulary continuous-speech. *IEEE Signal Processing Magazine*, *13*(5), 45-57.