JOHANNES KEPLER
UNIVERSITÄT LINZ | JKU

TNF

Technisch-Naturwissenschaftliche
Fakultät

Accurate Audio-to-Score Alignment – Data Acquisition in the Context of Computational Musicology

DISSERTATION

zur Erlangung des akademischen Grades

Doktor

im Doktoratsstudium der

Technischen Wissenschaften

Eingereicht von:
Dipl.-Ing. Mag. Bernhard Niedermayer

Angefertigt am:
Institut für Computational Perception

Beurteilung:
Univ.-Prof. Dipl.-Ing. Dr. Gerhard Widmer (Betreuung)
Dipl.-Ing. Dr. Alois Sontacchi

Linz, Februar 2012

Acknowledgements

First, I want to thank Gerhard Widmer. He did not only supervise this thesis, but also gave me the chance to work at his lab at the *Johannes Kepler University* of Linz. He was able to provide funding granted by the Austrian Science Fund (FWF) throughout the whole period of time I was working on this thesis, which is remarkable when taking the general economic situation during the last years into account. More precisely, the majority of the work presented here has been conducted within the scope of the projects *Computational Music Performance Research, Applied* (TRP 109-N23) and *Computational Performance Style Analysis from Audio Recordings* (P19349-N15). In addition, I want to thank Gerhard Widmer for maintaining a perfect balance between giving me the freedom to focus on research questions according to my own interests and to direct my efforts towards the respective project objectives.

Thanks in advance go to Alois Sontacchi for taking the time to assess this thesis and to act as examiner in its defense.

I also want to thank my colleagues at the lab, not only for the countless number of valuable discussions, but also for the great working atmosphere. It is a pleasure to work in an environment where, literally, all doors are open, and everyone is open to share one's thoughts one current research questions, latest results of one's own work, or simply a "nice-to-have" shell script.

I acknowledge my gratitude to Gerhard Widmer, Simon Dixon, and Werner Goebl who acquired the corpus of Mozart sonatas in the context of a project at OFAI (the *Austrian Research Institute for Artificial Intelligence* in Vienna). An adapted version of this data corpus is used for the evaluation throughout this thesis.

Finally, I want to thank my family for giving me the support and the environment to pursue an education at the university level which has now led to the completion of this thesis. Special thanks go to Birgit for accepting my increasingly long night shifts spent in front of screen and keyboard.

Kurzfassung

Die Aufgabenstellung die dieser Dissertation zu Grunde liegt ist, bestehende Systeme zum automatischen Synchronisieren von elektronischen Notentexten zu entsprechenden Audio-Aufnahmen so weit zu verbessern, dass sie zur halb-automatischen Extraktion der genauen Anschlagzeiten aller Noten in einem Klavierstück verwendet werden können. Der Fokus liegt dabei auf der Minimierung der Anzahl an Korrekturen, die ein Benutzer am Ergebnis vornehmen muss.

Bestehende Synchronisationssysteme behandeln eine beliebige Anzahl an Noten, die als gleichzeitig notiert sind, als ein einzelnes Ereignis. Die Konsequenz daraus ist, dass jede dieser Noten durch eine einheitliche Anschlagzeit beschrieben wird. Ein wesentlicher Beitrag dieser Arbeit ist es, Strategien zu Verfeinerung vorzustellen, die diese, musikalisch unbegründete, Vereinfachung beseitigen. Durch diesen Schritt wird die korrekte Behandlung von Asynchronizitäten und Akkord-Zerlegungen möglich.

Zur genauen Extraktion der Anschlagzeiten einzelner Noten wird ein Audio-Merkmal eingeführt, welches die Aktivierungsenergie einzelner Töne beschreibt. Es beruht auf einer modellbasierten Faktorisierung des Spektrogram und kann daher, im Gegensatz zu filterbasierten Methoden, an den musikalischen Kontext einer Note angepasst werden. Basierend auf diesen Techniken werden Anwendungen im Bereich der computergestützten Musikwissenschaft diskutiert. Dabei stellt ein Software-Werkzeug zum Inspizieren und Korrigieren automatisch erstellter Annotationen eine nötige Grundlage dar. Genaue Transkriptionen einer Interpretation können dann visualisiert oder zum Ableiten von Interpretations-Modellen verwendet werden.

Die Evaluierung der vorgestellten Methoden wird auf einem Korpus klassischer Klaviermusik durchgeführt. Dieser umfasst mehr als 100.000 Noten und eine Spielzeit von ungefähr vier Stunden. Durch den Einsatz eines computergesteuerten Konzert-Flügels konnten korrekte Annotationen zu den Audio-Aufnahmen gewonnen werden. Somit ist es möglich, die zeitliche Abweichung der automatisch extrahierten Anschlagzeit einer Note von der tatsächlichen zu messen. Im Rahmen dieser Arbeit ist es gelungen ein System zu entwickeln, bei dem der Median dieser Abweichungen über den gesamten Evaluierungskorpus kleiner als 10 Millisekunden ist.

Abstract

The objective of this thesis is to introduce enhancements to existing Audio-to-Score Alignment systems which allow for a semi-automatic annotation of note onsets within audio recordings of different performances of classical piano pieces. The main focus is put on minimizing the number of note onsets a human annotator would need to correct.

If an arbitrary number of notes is notated concurrently, it is treated as one single event by current state-of-the-art systems. Consequently, such a set of notes is assigned a common onset time. As one of the main contributions of this thesis, refinement strategies which resolve this simplification are proposed. This important step allows for dealing with stylistic details such as arpeggiations and asynchronies.

For the accurate extraction of an individual note's onset, a feature is introduced that represents the activation energy of a single pitch. It relies on a model-based factorization of the spectrogram and can, in contrast to filter based approaches, be tuned to the musical context of a note.

Based upon these techniques, a number of applications in the domain of computational musicology are discussed. Here, a prerequisite is an annotation tool, where the user can inspect and correct the automatically computed alignments. Accurate performance transcriptions can, then, be used for visualization of performer specific characteristics or the generation of respective performance models.

The evaluation of the proposed methods is performed on a corpus of classical piano music comprising more than 100,000 notes and a performance time of about four hours. An exact ground truth annotation corresponding to the audio recordings was obtained by using a computer controlled grand piano. This allows us to measure the deviation between the actual and the automatically extracted onset times. Within the scope of this thesis, a system was developed for which the median of these time deviations is below 10 milliseconds over the entire evaluation corpus.

Contents

List of Figures

List of Tables

1. Introduction

Music is the universal language of mankind.
Henry Wadsworth Longfellow, (in *Outre-Mer*)

1.1. Motivation

When listening to individual performances of a piece of classical music, it is likely that we hear differences in terms of dynamics, global and local tempo, or articulation. While it is relatively easy to get an intuition of a performer's style and to describe it qualitatively, it is difficult to obtain quantitative measurements of the expressive details of a performance. A major subfield of *Computational Musicology* concerns itself with the automated extraction and description of quantifiable aspects of musical performances.

To measure and understand how a performer uses the available degrees of freedom to give a piece a certain expression, a prerequisite is that those parameters, such as timing and loudness of individual notes, are known. A valuable source of such information are computer-monitored musical instruments. One example of a corpus obtained this way is the entire solo piano work of *Frédéric Chopin* performed by the Russian pianist *Nikita Magaloff* in 1989 on a Bösendorfer SE grand piano in Vienna. The resulting transcriptions of these performances, comprising precise measurements of every key and pedal action, have been extensively studied in the course of the *Magaloff Project* (see [Flossmann et al., 2010] for a report). Interesting findings do not only concern performance errors, between-hand asynchronies, and tempo rubato, but also the shaping of certain rhythmic passages.

However, transcriptions from computer-monitored instruments are rarely available for performances of famous artists. In contrast, audio recordings of professional performances can easily be obtained. The problem arising from using audio material as a data source is to extract an annotation at a level of accuracy which allows for the analysis of musical expression. Considering that the human auditory system is able to

recognize temporal displacements of down to a few milliseconds, it is obvious that a manual annotation of large corpora is very time-consuming and expensive.

An alternative would be the automatic transcription of recorded music. While this problem is considered to be solved for monophonic pieces, it remains an open problem for polyphonic music. In the context where different performances of a known piece are analyzed, however, one is not primarily interested in extracting the performed melodies or harmonies which are inherently determined by the score. Here, the problem of *Blind Audio Transcription* can be reduced to estimating the expressive parameters, such as relative loudness or articulation, for each note. Extracting each score note's timing in particular yields a synchronization between the score and the audio recording of a performance which is called *Audio-to-Score Alignment*. In comparison to audio transcription, state-of-the-art algorithms for such Audio-to-Score Alignments are relatively robust.

In order to be able to obtain annotations of recorded music which can be used in performance research, one has to achieve an adequate alignment accuracy. Therefore, the main focus of this thesis is placed on refining Audio-to-Score Alignments such that a human annotator's job can be reduced to the task of inspecting and correcting automatically generated timing annotations.

1.2. Objectives and Contribution

A long-term objective is to build an entirely automatic tool to align arbitrary audio recordings to their corresponding scores, including the extraction of the exact timing and loudness of each note while reporting playing errors or deviations from the score. However, while trained listeners are able to follow the score when listening to a performance or can even transcribe heard music, it is questionable if they are able to distinguish between different levels of loudness of individual notes of a chord. Furthermore, when looking at a sound's waveform, i.e., the representation of music available to computational processing, it seems to be impossible for humans to even tell apart musical content from speech, a mixture of both, or just noise.

MIR-research cannot per-se rely on high-level concepts as perceived by the human auditory system, such as pitch, note onsets, or timbre. Although considerable advances towards equaling the human perceptual capabilities have been achieved during the last 30 years, the extraction of each mid- or high-level representation of music is prone to errors or imprecisions. Therefore, even when the score of a performed piece is known,

accurately synchronizing it with a corresponding audio recording remains a difficult task.

1.2.1. Problem Description

This thesis concerns itself with Audio-to-Score Alignment in the context of classical piano music. Classical music is predestined for this task since, in general, the score as well as numerous performances by various pianists are known and publicly available. Piano is a musical instrument with limited degrees of freedom and therefore relatively easy to model. A note onset, for example, is always a sudden transient after which the loudness of the tone is subject to decay of the strings's oscillation and cannot be increased any more. Also, computer controlled mechanical instruments able to reproduce the full range of expressive variations are available and can be used to obtain highly accurate ground-truth data. The piano is therefore the subject of numerous MIR research activities.

The resulting alignment system is intended to be used as an annotation tool. For each note of the score, the exact onset time within a corresponding audio recording has to be measured while requiring as few user interactions (i.e., manual corrections) as possible. Hence, the critical factor is the accuracy of note onsets, while other aspects such as note durations or their loudnesses are neglected. The objective is to maximize the number of notes where the deviation of the extracted onset time from the real one is not noticeable. It is desirable that notes where this goal is not attained, are still aligned within a second, wider tolerance range. However, since in such cases a manual correction is required anyway, the overall robustness with respect to such a threshold is only a secondary objective in this context.

A second auxiliary goal is to keep computational costs low. The task at hand is an optimization problem, where many of the algorithms used have an asymptotic complexity of $O(n^2)$ or even worse in space as well as in time. Increasing accuracy by providing additional features or features calculated at higher resolutions is limited by such computational constraints.

One possible approach for Audio-to-Score Alignment is to extract the same features as calculated from the audio recording from the score as well. In such cases, the problem at hand is an Audio-to-Audio Alignment. Due to the inherent similarities between these two subfields, the term *audio alignment* will be used in situations where a discrimination is not necessary.

1.2.2. Contributions

While the original ideas of audio alignment and also its online correspondent, score following, date back several decades, the contributions of this thesis comprise several new approaches and improvements of existing methods concerning all stages of the alignment process.

Feature Extraction A new audio feature – Pitch Activation – which can be understood as a rudimentary multiple pitch estimator is introduced. It is calculated by factorizing a spectrogram using pre-trained tone models and a non-negativity constraint. In analogy to a (comb-)filter bank it also accounts for a fundamental frequency's harmonics. However, in contrast to filter banks, the musical context can be taken into account by training the tone models on a certain instrument and considering only those models corresponding to notes which are expected to be played within a certain time window under consideration. The Pitch Activation feature will be described in Section 3.4.

Global Alignment The literature describes two approaches for the alignment between the symbolic score and audio recordings. One is based on decoding a graphical model of the score when certain audio data is observed. The other one, extracts audio features from the score, either directly or by synthesizing it, yielding a second audio recording. Here, the third alternative, next to these audio-only and mixed domain approaches, is explored. Based on a quasi-transcription of the audio recording, the actual alignment is performed in the symbolic domain. Although we found that this method cannot compete with state-of-the-art systems in terms of accuracy, it achieves a significant data reduction and is, therefore, valuable for the processing of very long pieces. This system will be described in Section 4.3.

Aiming at computational efficiency as well, we also describe an automated *divide & conquer* version of the Dynamic Time Warping algorithm. Based on an initial alignment calculated at low feature resolutions, notes are identified for which an exact timing can be extracted with a high confidence. Then the algorithm – which is of complexity $O(n^2)$ – is performed at higher feature resolutions on each individual segment between two such notes. This method will be described in Section 5.1.6.

Note-level Alignment A vast majority of state-of-the-art approaches to Audio-to-Score Alignment considers all events notated concurrently – single notes as well as whole chords – as one single object. Hence, they are matched to the same timestamp within the audio recording. In the domain of computational musicology and performance analysis such as conducted in the context of the *Magaloff project*

(cf. [Flossmann et al., 2010]), however, exact timing information is required for each individual note. Therefore, post-processing methods aimed at refining each note's onset time individually are proposed. This is the main contribution of this thesis and will be described in Chapter 6.

Evaluation We evaluate the described algorithms using various configurations on a unique dataset comprising more than 100.000 notes. The data was obtained from a live performance of a professional pianist on a computer monitored grand piano, yielding the exact parameters for each played note. This allows for large scale evaluation at the note-level. Also, we present a detailed analysis of audio material produced by software synthesizers to give us an idea about how our results compare to those of other authors using such audio sources. Evaluation issues will be discussed in Chapter 2.

Applications Finally, an annotation tool incorporating an automatic Audio-to-Score Alignment functionality is presented in Section 7.1. For the user's convenience, he can choose between a fast and computationally efficient algorithm and our proposed system which is tuned towards high accuracy. We also demonstrate the use of the acquired data as well as the application of developed algorithms to related tasks in the context of computational musicology, such as the automatic extraction of tempo curves and corresponding visualizations, structural analysis, and version detection. These applications will be described in Chapter 7.

1.3. Outline

This thesis is structured as follows. In Chapter 2, the evaluation of Audio-to-Score Alignment is discussed. There, not only the used data corpus as well as chosen evaluation metrics and adequate result plots are described. An issue which is also covered is the comparability of results obtained from different audio material, such as synthesized MIDI in general, or data synthesized from symbolic music representations covering expressive variations in varying richness. Experiments are conducted that show that when using synthesized data for audio alignment experiments, natural micro-timings are most crucial to obtain meaningful results.

Chapter 3 concerns itself with the feature extraction process. Since a vast majority of current alignment systems rely on audio features computed in the spectral domain, transforms of the audio signal into a time-frequency representation which are relevant in this context are reviewed. Based upon these foundations, state-of-the-art features are

explained as well as the Pitch Activation feature computed by spectrogram factorization proposed by us.

According to the processing chain, Chapter 4 focuses on alignment techniques. Here, Dynamic Time Warping (DTW) and Hidden Markov Models (HMMs) are the methods commonly used the literature. In addition to these approaches, where the first works in the domain of audio features and the latter relates audio features to information in a symbolic representation, we present and evaluate a novel alignment system where both the audio and the score data are represented in the symbolic domain.

For reasons explained in Section 4.5 we decided to focus on the Dynamic Time Warping approach for Audio-to-Score Alignment. In Chapter 5, we, therefore, present a number of strategies optimizing DTW towards computational costs, robustness, and accuracy. Examples of such approaches are the divide & conquer method, our automatically obtained plausibility measure, and multi-scale DTW.

The main contribution of this thesis is the subject of Chapter 6. There, we propose a multi-pass system for the accurate Audio-to-Score Alignment. A main difference to the vast majority of related systems is that based upon an initial alignment each note is refined individually, such that intra-chord timing deviations can be resolved. We show that all our refinement steps improve the result for each individual piece of our evaluation corpus and that our refinement method outperforms a reference system.

Audio-to-Score Alignments can be considered to be annotations of audio recordings of specific musical performances. Such annotations are a valuable data source for musicologists. In Chapter 7 an alignment editor is introduced, where a computed alignment can be compared to the respective spectrogram in order to allow for manual corrections. Assuming correct as well as accurate alignments, visualizations of performance aspects can be computed. To show the versatility of the techniques described earlier, their application to tasks different from audio alignment is discussed. One example of such a task is the identification of versions of the same piece of music played by different historical automata (e.g., flute works or musical boxes). Another example is the automatic recognition of a song's structure.

Chapter 8 concludes this thesis. In addition to a discussion of insights gained throughout this work, some ideas for future research activities are sketched.

2. Evaluation

The focus of this thesis is the development of accurate Audio-to-Score Alignment methods. In order to be able to measure if and to which extent this objective is achieved, a systematic evaluation is essential. In this chapter an adequate framework will be established such that each of the later described approaches can be benchmarked on an equal basis. The test data comprise audio recordings of classical piano music played on a computer-monitored grand piano. Thus, exact symbolic transcriptions are available and are used as ground truth data. Further, meaningful evaluation criteria are chosen. At the end of this chapter, alternative data sources are reviewed.

The detailed discussion on test data and evaluation methods is inevitable, since the MIR community has not yet established a common evaluation model and an according data set for (offline) Audio-to-Score Alignment. While open evaluation campaigns had already been established in fields such as text based information retrieval, the *ISMIR 2001 Resolution on the Need to Create Standardized MIR Test Collections, Tasks, and Evaluation Metrics for MIR Research and Development* was not drafted before 2001 [Urbano, 2011]. In 2005, the first *Music Information Retrieval Evaluation eXchange* (MIREX)[1] was carried out. Although the campaign's remarkable value to the research community is out of question, the selection of evaluation tasks is limited.

Audio-to-Score Alignment was proposed as a MIREX task in 2006, but has been dropped and never considered again. However, the closely related topic *Real-Time Audio-to-Score Alignment (a.k.a. Score Following)* was part of the 2006 evaluations and is still on the schedule for the MIREX 2011. From this fact, one can eliminate a lack of royalty-free audio material including ground truth annotations as a possible reason for the disregard of offline Audio-to-Score Alignment as a MIREX task. One can argue that annotations at the chord level – as used for Score Following – are not sufficiently accurate for the evaluation of offline alignment results. Nevertheless, in the author's opinion, offline Audio-to-Score Alignment is not carried out as a MIREX task due to a low community interest for a uniform evaluation. A plausible reason are the divergent objectives behind audio alignment research. While this thesis focuses on alignment accuracy within a constrained environment, other authors aim for ro-

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

bustness or computational efficiency, such as [Müller et al., 2004], [Müller et al., 2006], [Pardo and Sanghi, 2005], [Raphael, 2001], or [Salvador and Chan, 2004].

2.1. Data Corpus

Computational music perception heavily relies on methods from Artificial Intelligence and Machine Learning. In these fields, a typical evaluation procedure is based on multiple independent data sets. A classifier is initially adapted to a given training set. To avoid over-fitting and to achieve the ability to generalize, an additional validation set is used to terminate learning algorithms as soon as the classification performance on these unseen data decreases. Another approach to obtain a classifier with a good generalization behavior is to execute pruning as a post-processing step. Finally, the resulting classifier is evaluated on a test set, which is not involved in any phase of the training.

In this thesis, however, instead of applying machine learning algorithms, algorithms and models are developed based on musically well-founded assumptions. Therefore, there is no necessity for separate training and validation sets and the evaluation of the described methods can be performed on the sole basis of one corpus of test data.

The chosen genre is classical piano music, for several reasons. On the one hand, classical music is highly relevant for Audio-to-Score Alignment from an applications point of view. A set of well known pieces – where the scores are generally available – is performed by numerous artists, each of them exhibiting an individual style. The work described here was conducted within the context of two projects concerned with investigations into such expressive elements. Detailed and accurate annotations are an essential prerequisite for the analysis and description of different interpretations of a piece and the respective expressive details. However, symbolic records of performances by great artists are notably rare, which calls for a means to derive such data from commonly available audio material.

On the other hand, audio transcription and similar tasks, such as Audio-to-Score Alignment, for polyphonic pieces are an open research problem. The fundamental difficulty is to identify individual partials within a complex audio signal and correctly relate them to a certain pitch. From a different point of view, this task can be reduced to deciding which combination of pitches is the most likely cause for a certain observation of weighted partials. Due to overlapping partials, inharmonicity of musical instruments, and the presence of noise, this task is highly complex and error prone. A common

strategy is therefore to restrict the audio material to recordings of one certain type of instrument, thus avoiding the additional problems of different timbres.

In the context of musical signal processing, restrictions towards the piano are a common choice (see [Abdallah and Plumbley, 2004], [Arifi et al., 2003], [Dixon, 2000], [Dressein et al., 2010], [Poliner and Ellis, 2007], or [Schwarz et al., 2004], for example). Due to limited expressive freedom, it produces a relatively consistent sound. Note onsets, for example, are inherently characterized by a transient attack (in contrast to 'soft' onsets of string or wind instruments where the signal energy can increase continuously during a certain span of time). Such an unambiguous onset event is not only relevant for musical signal processing systems but also for their evaluation.

2.1.1. The Bösendorfer SE 290

Although the grand piano *Model 290 "Imperial"*[2] was first built around 1900, it is still Bösendorfer's top model. With its length of 290 cm it is also the company's most voluminous grand piano. Additionally, distinguishing it from other grand pianos, it has a range of 97 tones (from C0, i.e., MIDI pitch 12, to C8, i.e., MIDI pitch 108), covering eight full octaves.

The SE recording and reproduction technology was developed by the engineer Wayne Stahnke, who is also acknowledged in the name SE – i.e., *Stahnke Electronics*. Bösendorfer licensed the system and delivered the first prototype to the MIT Artificial Intelligence Laboratory in 1985. A year later, the SE system for the Model 290 was launched as an official product.

The recording technology is based on optical sensors – i.e., pairs of fixed LEDs and photo-transistors in combination with aluminum shutters attached to the keys and hammers. The adjustment is such that a key event is triggered the instant the corresponding key is pressed perceptibly (having a play of about 2 mm). The shutter mounted at the hammer has an arm with a width of only a few millimeters which passes the beam of light between LED and photo-transistor before the shutter itself discontinues the lighting after a hammer movement of exactly 5 mm, thus providing two events. The setup is such that the second event occurs the moment the hammer is about to hit the strings. The sensor's values are sampled using a frequency of 800 Hz, yielding a time resolution of 1.25 ms. While the event when the hammer hits the strings is taken as the note onset, the note offset is determined by the key release. Pedal positions are also measured at lower time resolutions and quantized using a range of 256 values (cf. [Goebl, 2003] and [Moog and Rhea, 1990]).

[2] http://www.boesendorfer.com/en/model-290-imperial.html

The sensor information is not only used to determine the exact note timings, but also to calculate the *final hammer velocity* (FHV) based on the time interval between the two hammer events. For this purpose a much higher sampling rate of 25.6 kHz is used. The number of samples it took the hammer to cover these final 5 mm is defined to be the *inverse hammer velocity* (IHV). The FHV in $[m/s]$ is then obtained as

$$FHV = 25600 \times 0.005 \times \frac{1}{IHV} = \frac{128}{IHV} \tag{2.1}$$

In order to be represented in MIDI format, from the IHV as stored in Bösendorfer's proprietary file format, corresponding key velocitys v_{key_n} have to be obtained. To this end the FHV is logarithmically mapped while also considering a pitch dependency, as

$$v_{key_n}(FHV) = 52 + 25\log_2(FHV) + \frac{n-60}{12} \tag{2.2}$$

where n is the MIDI note number. An FHV value of $4.26\,m/s$ – which is an upper limit for a typical piano performance (see [Goebl, 2001]) – translates to a MIDI velocity of 104. The lowest possible hammer speed, corresponding to a MIDI velocity of 0, is $0.236\,m/s$. This accounts for the fact that if the hammer velocity is below a certain threshold, the hammer will not reach the strings any more, resulting in a silent note.

The reproduction on the SE system relies on special solenoids specific to the speed, accuracy, and size requirements. Each key has its own actuator attached in a way such that it only interferes with the key while the instrument is in playback mode. The same holds for the pedals. The controller and its firmware are responsible to account for the variable time delay between the initial key actuation and final hammer strike which can account for as much as 100 ms [Moog and Rhea, 1990][Goebl, 2003].

In [Goebl and Bresin, 2003] and [Goebl, 2003], the recording and reproduction accuracies of the Bösendorfer SE 290 as well as the Yamaha Disklavier were studied. It is shown that both systems suffer from a systematic error concerning clock timing. The result is that when recording, there is a linearly increasing anticipation of events, i.e., a time compression. However, this timing error is canceled out during playback. The residual reproduction error after a recording-playback cycle was found to be smaller for the Bösendorfer SE system. The mean time deviation was 0.2 ms at a standard deviation of 2.1 ms, while the corresponding values for the Yamaha Disklavier were 1.4 ms and 3.8 ms respectively. It also became apparent that the SE system tends to recognize louder tones later than softer ones. This tendency is reversed during reproduction, where a mean timing deviation of -0.1 ms at a standard deviation of 1.3 ms was ob-

served (as opposed to -0.3 ms and 5.5 ms for the Disklavier). In general, Bösendorfer's SE system is significantly more robust to outliers than the Disklavier. The complete recording and reproduction cycle produced only very rare timing errors larger than ±3 ms, while the largest time displacements on the Disklavier are almost ten times as large. Concerning dynamic accuracy, the SE system outperformed the Disklavier as well. This is due to the Disklavier's inability to reproduce very loud as well as very soft tones – an effect that could not be observed at the Bösendorfer piano.

In general, the Bösendorfer SE 290 is a high end grand piano featuring an accurate recording and reproduction technology introducing timing errors of only a very small number of milliseconds. Its output is in a proprietary file format, which can, however, be converted to the MIDI format. Since 2005, Bösendorfer has been shipping the next generation model of its reproduction system under the name *Ceus*[3]. While the basic functionality of this system equals that of the SE system, the company claims that the upper limit for reproduction timing errors has decreased from about 3 ms to 2 ms.

2.1.2. The Mozart Sonatas

The data corpus used throughout this thesis comprises performances of the Fantasia in c minor (K. 475) and 12 out of the 18 piano sonatas by *Wolfgang Amadeus Mozart*. Since the fantasia was published in conjunction with the 14^{th} sonata (c minor, K. 457) we will refer to the whole set of pieces as the *Mozart sonatas*. They were composed in the years between 1775 and 1788 – i.e., during the Classical era. All of them are consistent with the characteristic form having three movements.

The recordings originate from a performance of the complete Mozart piano sonatas by Roland Batik – a professional Viennese pianist – on a Bösendorfer SE 275 in 1990. The internal Bösendorfer performance files were converted into MIDI as described above. Also, corresponding audio material was obtained from a playback made by Werner Goebl on an SE 290 at the Bösendorfer company in 2001. Recordings were made at 44.1 kHz using a single high-quality microphone near the corpus of the piano and a DAT recorder. An overview of the sonatas, listing the number of score notes and the respective performance time for the individual movements, is given in Table 2.1.

This material is of special value, since it does not only describe the pianist's actions including note timings, dynamics, and also the exact pedal pressure at any time during the performance, but also the corresponding audio recording are obtained from a high end concert piano. They are, therefore, subject to effects such as room acoustics or oscillations of the instrument body.

[3] http://www.boesendorfer.com/en/ceus-reproducing-system.html

piece	1^{st} movement		2^{nd} movement		3^{rd} movement		all	
	notes	time	notes	time	notes	time	notes	time
k.279	2803	4:55	1705	6:49	2890	4:36	7398	16:20
k.280	2497	4:48	1141	5:41	2442	4:33	6080	15:02
k.281	2651	4:29	1522	5:42	2228	4:26	6401	14:37
k.282	1899	7:35	1744	4:22	1928	3:02	5571	14:59
k.283	3304	5:22	1983	8:08	2602	4:06	7889	17:36
k.284	3707	5:17	1506	4:56	7565	15:55	12778	26:08
k.330	3160	6:14	1456	6:40	2977	5:43	7593	18:37
k.331	6160	13:35	2673	5:50	2804	3:26	11637	22:51
k.332	3474	6:02	1278	5:04	3997	6:56	8749	18:02
k.333	3782	6:44	1978	7:22	3090	6:29	8850	20:35
k.457	3003	6:15	1750	7:30	2175	4:35	6928	18:20
k.475	1308	4:58	1318	3:23	1276	3:44	3902	12:05
k.533	4348	8:25	1601	6:58	2678	7:01	8627	22:24
all							102403	3.57:36

Table 2.1.: Performance time and number of score notes (repeated sections are counted twice) of the 13 Mozart sonatas used for evaluation throughout this thesis

2.2. Test Bench Environment

The Audio-to-Score Alignment systems examined here, require a MIDI file representing the score (referred to as *Score MIDI*) and a monaural audio recording sampled at 44.1 kHz in waveform format (wav). The output is a MIDI file which is time aligned to the performance. This result (referred to as *Performance MIDI*) can then be compared to the ground truth data obtained from the Bösendorfer SE system. However, due to expressive variations and playing errors, the notes played, i.e., MIDI events contained in the performance data, do not precisely match the score. As described in [Flossmann et al., 2010], this makes an automatic matching error-prone, even if it is carried out in the symbolic domain.

2.2.1. Matching Score MIDI to Performance MIDI

Scores for the Mozart sonatas are publicly available in numerous editions. A well known one, which is offered online and free of charge for personal study and educational use, is the *Digital Mozart Edition*[4] by the *Internationale Stiftung Mozarteum*. For the task of Audio-to-Score Alignment itself, one would derive the score MIDIs from such an edition. However, for the evaluation of alignment results, for each score note the corresponding events in the performance MIDI have to be identified. The parameters of these events

[4] http://dme.mozarteum.at/DME/nma/start.php?l=2

are the target values which an alignment or annotation system is expected to extract, i.e., the ground truth.

Assuming a performance without note insertions, deletions, or substitutions, the matching between a score MIDI and the performance MIDI could be easily done by relating notes of the same pitch to each other based on their temporal order. However, in a real performance of a demanding piece of music, the artist will most likely introduce expressive variations and make some playing errors, i.e., play incorrect notes, leave some notes out, or insert other ones. The classification of such variations is ambiguous in numerous cases, making an automatic matching between a score and the performed notes error-prone.

error	**1st movement**		**2nd movement**		**3rd movement**		**all**	
	count	**rel**	**count**	**rel**	**count**	**rel**	**cnt**	**rel**
Insertions	1411	3.3%	1414	6.1%	1367	3.4%	4192	3.9%
Deletions	92	0.2%	20	0.1%	84	0.2%	196	0.2%

Table 2.2.: Playing errors made during the performance of the Mozart sonatas (wrong notes are represented as a deletion of the correct note plus an insertion of the wrong note). The percentage of insertions is calculated relative to the number of performed notes while the fraction of deletions is relative to the score notes.

As can be seen from Table 2.2, there are almost 4500 errors when counting a wrong note as both, the deletion of the correct note and the insertion of a different one. While there are relatively few deletions, the number of insertions is surprisingly high. Further examination showed that this effect is not caused by insertions of individual notes at a time, but by introducing bridge passages or additional trills.

Another interesting observation in this context is that the number of deletions is significantly lower for the second movements. The slower tempo of these movements leads to the assumption that the pianist might have dropped notes during fast and technically challenging passages in order to keep the pace. The average tempos, given in beats per minute, over an entire movement of a sonata are shown in Table A.1 – Table A.3. There, for movements with a time signature of 2/2 (*alla breve*), the quarter note is considered to be the beat. In doing so, the resulting median tempo is 126 bpm and 138 bpm for the first and the third movements respectively, while this value is only 52 bpm for the second movements.

For evaluation purpose, the correct correspondence between score notes and performed notes, i.e., the target values an alignment system is expected to extract , as well as the classification of deviations as insertions or deletions of certain notes had been corrected manually and was stored to files for later usage. Also, erroneous events reported by

the SE system which were due to hammer bounces were removed from the performance records. This work was not only conducted by the author but mainly by Gerhard Widmer, Werner Goebl, and Simon Dixon. The file format used to store the matches between score and performance notes is purely text-based. Each matched pair is represented as a description of the score note, including, amongst other data, an ID, its pitch, score time, and note value, and information about the performed note, such as its exact timing, loudness, and duration. In addition, meta information can be specified, e.g., respective score, MIDI, and audio files, the musical key, or the time signature.

2.2.2. Matching Performance MIDI to Audio

The performance MIDI serves as the ground truth transcription of a performance. However, MIDI data converted from Bösendorfer's internal format suffers from the systematic timing error of the SE system, as reported in [Goebl and Bresin, 2003]. In addition, there is an offset between the beginning of the audio recording and the playback on the Bösendorfer SE 290.

To resolve these problems, a manual detection of the first onset within each audio file was performed to eliminate the offset between the beginning of the audio recording and the start of the playback. Furthermore, with regard to audio recordings as commonly available on the retail market, we treated the individual movements of the sonatas separately.

The general timing error was corrected by performing a linear time stretch. To this end, the onset time of the last notes were manually detected as well and the MIDI times were scaled accordingly.

An overview of the resulting test bench environment is presented in Figure 2.1. The score MIDI, the performance MIDI, i.e., the ground truth annotation of a certain performance, and the correspondence between individual notes within those two records are stored within a single file. The Audio-to-Score Alignment system is then executed (upper part of Figure 2.1). It uses the score MIDI and the audio recording of a corresponding performance to compute an estimation of the performance MIDI (blue). At the evaluation step (lower part of Figure 2.1), this result is compared to the performance MIDI, i.e., the ground truth data, which is not presented to the system until then. At this stage, the pre-established match between score notes and the performance MIDI is used, such that for each aligned note its respective target parameters are known. To guarantee that this correspondence is not lost during the alignment step, unique identifiers are assigned to the score notes beforehand.

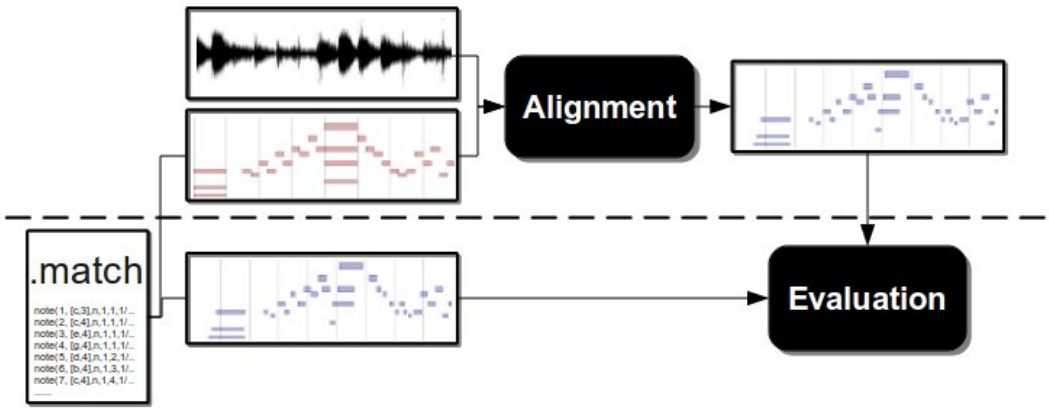


Figure 2.1.: Overview of the test bench used for the evaluation of Audio-to-Score Alignment results: The upper part shows the alignment step. Here, only the score MIDI (red) and the audio signal are shown to the Audio-to-Score Alignment system. The performance MIDI and the correspondence between score and performance MIDI are only known to the evaluation system which is shown in the lower part. Here, the result of the alignment system can be compared to the ground truth.

2.3. Evaluation Criteria

Given an annotated data corpus and a task specific test bench, appropriate evaluation criteria have to be selected. Adequate measures are a prerequisite in order to achieve validity of an evaluation experiment. *Criterion validity* itself aside, [Urbano, 2011] describes *construct validity* as the requirement of the measured variables to closely correspond to the concept they are supposed to quantify. A similar demand, that those variables are not influenced by factors unaccounted for is named *internal validity*. Also, a criterion should agree with other measures it is, in theory, related to (*convergent validity*) and allow for justified conclusions (*conclusion validity*). (For a detailed discussion on MIR meta-evaluation in general the reader is referred to [Urbano, 2011].)

Validity aspects have been taken into account when discussing the following evaluation criteria. Although this thesis focuses on timing accuracy, measures aiming at other aspects, such as loudness of individual notes, are briefly introduced.

2.3.1. Timing

Audio-to-Score Alignment can be described as identifying each score note within an audio recording. In doing so, the accurate extraction of the notes' onset times is essential. On the one hand, inter-onset intervals play a major role in the human perception of music. [Friberg and Sundberg, 1992] show that within a series of notes played at a

certain frequency, timing displacements of down to a few milliseconds can be recognized by human listeners. On the other hand, a correct onset time is a prerequisite for the extraction of other note parameters, such as duration or loudness.

Graphical Representation

In general, timing accuracy is measured in terms of (absolute) timing deviations between the actual onset times and the onset times reported by the alignment system. The resulting values can be plotted against the notes or on the performance time as proposed by [Müller, 2011]. The advantage of this presentation of results is that passages showing large time deviations become obvious and can be directly accessed in the score and the audio files respectively. In addition, not only can verified ground truth data serve as basis for an evaluation, but a result can also be compared to an alignment obtained using a different algorithm.

We propose two extensions to this representation. First, we add a histogram showing the distribution of time displacement values. The motivation is that while the deviation curves clearly exhibit general trends, i.e., sections of accurate or failed alignment, the overall quality of an alignment does not become apparent. The histogram, in contrast, shows not only the variance of displacements, but also if an algorithm generally tends to predict note onsets too early or too late.

Secondly, assuming the score is presented to the system in a proper format, the time interval measured in the unit of beats between arbitrary pairs of notes can be determined. From this information, local tempo estimates can be calculated for an alignment. Since the same absolute time displacement is more noticeable during a playback of fast passages, we propose a tempo-relative deviation measure in the unit of score time – i.e., beats. The resulting values do not only reflect on how similar a playback of the aligned score would sound to the original audio recording, but also allow for error analysis in the score domain.

An example of such a plot is shown in Figure 2.2. The plot on the left shows the timing error for each note measured in beats, i.e., a relative tempo unit. The histogram on the right shows the distribution over the magnitudes of timing deviations which will, in most cases, approximate a Gaussian. However, by observing the mean and the modus, one can draw conclusions about the tendency of an algorithm to report note onsets early or delayed. In addition, the cumulative distribution function of the timing errors is plotted. Since the maximum of this function is a known value, the plot is scaled such that entire available space is used.

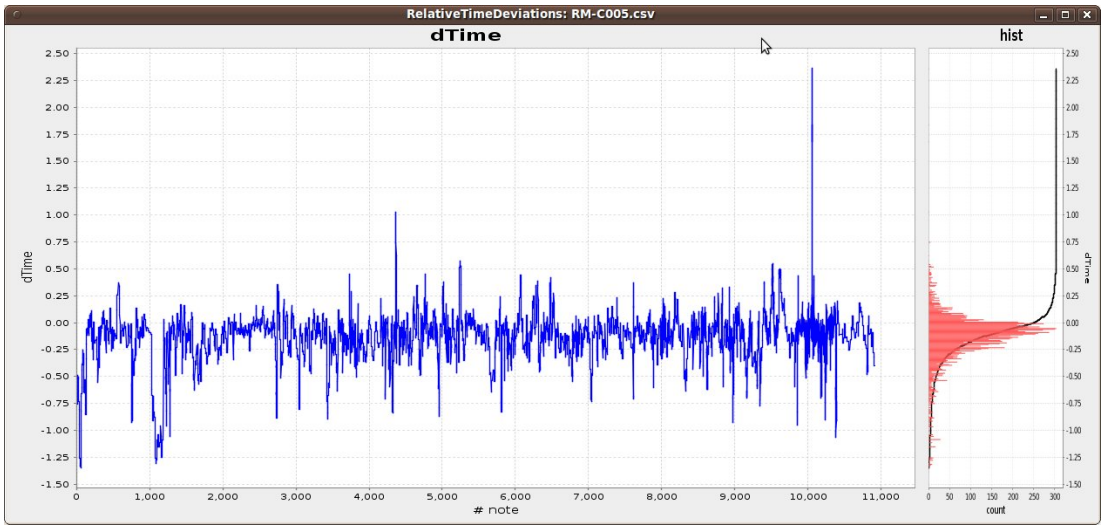


Figure 2.2.: Timing error of an example alignment measured in terms of beats

Average Time Displacement

More compact than graphical representations, are statistical figures. The probably most obvious ones are the mean time displacement and the corresponding standard deviation. Calculated on the signed values, the mean time deviation shows to which extent an aligned score runs ahead or behind the actual performance. Considering the absolute values, the result is the average error. This evaluation criterion is commonly used in the literature – see [Dannenberg and Hu, 2003], [Dixon, 2005a], or [Ewert and Müller, 2009], for example.

Also, from 2006 – where the closely related task of online Audio-to-Score Alignment, i.e., score following, was carried out as part of MIREX for the first time – the average timing error and the respective variance were used as assessment metrics for the evaluation campaign [Cont et al., 2007].

Methods to obtain criteria more robust to outliers, are to disregard the highest 10% amongst all timing deviations or to consider alignments of sections where the time displacement exceeds a certain threshold to be failed and evaluate such segments separately. An evaluation based on such a criterion can be found in [Niedermayer, 2009b], for example.

Percentiles

A major drawback of averaging errors is that the result is not robust to even relatively small numbers of outliers. Systems yielding a majority of highly accurate alignments

and a few outliers – such as an alignment failure at the end of a piece – might not be distinguishable from algorithms of a generally mediocre alignment accuracy. Assessing alignments by means of percentiles overcomes this flaw.

Calculating the median instead of a mean timing deviation makes the evaluation criterion robust to outliers. It is therefore an alternative to the mean of the best 90% of all timing deviations or the mean over "succeeded" alignments. However, in the author's opinion, evaluation based on the median timing deviation is more clean from a methodical point of view, since it does not involve the arbitrary selection of a cutoff ratio. The median time deviation criterion is also, for example, used in [Dixon and Widmer, 2005] or [Shalev-Schwartz et al., 2004].

In addition to the median, we propose to also consider the 75^{th} and the 95^{th} percentile as well as the maximum. Those criteria give a better understanding of the time displacement behavior and robustness of a system. The maximum time displacement is likely to occur either at the end of the piece or in conjunction with long pauses where the alignment fails at a local level due to a lack of additional cues about the musical context. Therefore, only a very small number of notes is affected by such errors. Under these premises, the percentiles illustrate a soft error margin, which is not exceeded by the majority of time displacement values.

Tolerance Ranges

An alternative to calculating statistics on time deviation values, is to measure the number of events aligned more accurately than a predetermined threshold. Such a tolerance range was used in the MIREX 2006[5] evaluation of score following. There, notes with an absolute offset (in the context of score following the *offset* denotes the timing deviation plus the latency of the system) greater than 2000 ms were assessed as missed notes. Later an absolute error threshold of 300 ms was proposed by [Cont et al., 2007].

Nevertheless, even the tighter tolerance range of 300 ms is too indulgent. A more adequate threshold is the 50 ms range, well known from the field of Onset Detection (see [Bello Correa et al., 2005], [Dixon, 2006], or [Eyben et al., 2010], for example). An even stricter evaluation basis is an allowed timing error of 10 ms initially proposed in [Niedermayer, 2009a]. This range is motivated by the human ability to recognize time deviations of tones played at a constant tempo. In [Friberg and Sundberg, 1992] this just noticeable difference was found to be 10 ms for short inter-onset intervals and about 5% of the note duration for notes longer than 240 ms.

[5] http://www.music-ir.org/mirex/wiki/2006:Score_Following_Proposal

The tolerance range of 10 ms is of particular interest when the objective is to develop a semi-automatic Audio-to-Score Alignment system where the user manually corrects errors, i.e., "too large" timing deviations, within a generated alignment. In this scenario, each note with an alignment error not recognizable by a human listener would not require correction of its onset time. Misaligned notes, in contrast, would need to be post-processed independent of the actual value of error, which becomes secondary in this context.

An extension to this approach is used for the evaluation in [Dixon and Widmer, 2005], where the percentage of notes aligned more accurately than an iteratively increasing threshold is presented. Starting from the number of notes aligned to the exactly matching audio frame, the tolerance range in terms of frames is increased up to a span which covers 99.9% of all notes. A similar evaluation method is also applied in [Macrae and Dixon, 2010], where, in the context of online, real-time alignment, larger tolerance thresholds between 100 ms and 2000 ms were used.

2.3.2. Loudness

While timing accuracy is a well studied aspect of audio alignment, the automatic extraction of dynamics information is widely neglected. Although this is not an issue specific to Audio-to-Score Alignment, but also relevant to the field of audio transcription, Scheirer (see [Scheirer, 1997]) is the only author describing an approach for the extraction and evaluation of loudness information from audio signals for individual notes. MIDI velocities were estimated from the energy within the frequency bands relating to the fundamental and the harmonic frequencies of a certain note. The evaluation criterion applied was the correlation between the estimates MIDI velocities and the corresponding ground truth data obtained from a computer-monitored Yamaha Disklavier.

2.3.3. Other Evaluation Criteria

Timing and dynamics are essential aspects in musicology and performance research. However, this is not the only information contained in audio signals. Note durations and the use of pedals, playing style and articulation, or timbre of the specific musical instruments used for a performance are other details one might be interested in. Nevertheless, these issues will not be addressed here.

An aspect which, in contrast, affects each automatic system is computational efficiency. There, several approaches have been described and also evaluated in the lit-

erature. The efficiency of an algorithm is measured by means of asymptotic complexity, fraction of theoretically possible pairwise comparisons which had to be made, or computational speedup (see [Niedermayer, 2009b], [Niedermayer and Widmer, 2010b], [Müller et al., 2004], [Müller et al., 2006], or [Salvador and Chan, 2007], for example).

2.4. A Detailed Analysis of Possible Data Sources

Due to the lack of a publicly available test database suitable for the in-depth evaluation of Audio-to-Score Alignment, a large variety of different data corpora is described in the literature. In addition to the Bösendorfer SE system, the Yamaha Disklavier and hardware as well as software synthesizers are commonly used for data acquisition. However, not only the source of audio material differs between individual publications, but also the richness of expressive variations as a crucial element of a natural musical performance. In [Niedermayer et al., 2011a], we presented a study on the effect of these different sources of evaluation data on the actual results and their comparability. The main results are recapitulated in the remainder of this chapter.

2.4.1. Alternative Audio Sources

Audio-to-Score Alignment is a task which requires an evaluation at the note level in order to draw an accurate picture of an alignment system. However, corpora such as the one described and used here are rarely available and depend on the access to expensive computer controlled musical instruments (already assuming that such a computer monitoring system exists at all for a certain instrument).

Considering the vast variety of musical performances by world class artists which are (commercially) available, an obvious method to obtain performance data would be to manually annotate arbitrary recordings. However, this approach is not feasible for two reasons. On the one hand, manual annotations are very expensive. Even trained listeners will most likely need to hear each single note several times in order to be certain about how to set the parameters. One should bear in mind that timing deviations, for example, are measured at the millisecond level. On the other hand, there is reasonable doubt if certain expressive nuances can be distinguished by human listeners at all. Examples are intra-chord micro timings, i.e., asynchronies and arpeggiations, or the exact loudnesses of individual chord notes.

Considering the yearly held MIREX[6] [Downie et al., 2010] algorithm evaluation, there is a clear preference towards human annotated ground truth data for note- or beat-level evaluation tasks. The testing of systems in the domains of *Audio Onset Detection*, *Real-Time Audio-to-Score Alignment*, *Audio Melody Extraction*, *Audio Chord Estimation*, and *Audio Beat Tracking* solely relies on manually annotated data. Only *Multiple f_0 Estimation and Tracking* is evaluated based on a corpus also comprising recordings from a Yamaha Disklavier and synthesized audio.

Synthesizing audio from a given performance representation, i.e., a known ground truth, is the third approach to yield adequate data corpora. Respective performance MIDI files can be obtained from MIDI instruments or found on the Internet. In contrast to manual annotations, highly accurate data can be generated at low to moderate costs. However, it is not clear how natural and lifelike synthesizer-generated audio material will be and if such an evaluation base will interfere with the results.

We, therefore, examined audio data originating from two different software synthesizers. To this end an Audio-to-Score Alignment was performed on the respective audio signals and also on the recordings from the Bösendorfer SE 290. The evaluation results were then compared to each other.

Timidity

The first examined synthesizer was *timdity++*[7] (referred to as *timidity*) by Masanao Izumo. Versions for various platforms are distributed under the *GNU General Public License*[8]. Due to this availability free of charge, it is a common choice in the MIR literature (see [Hu et al., 2003], [Dannenberg and Hu, 2003], and [Tzanetakis et al., 2003] for example).

Timidity can generate audio using arbitrary sound fonts in GUS/patch format. Per default it is delivered in combination with a set of free instrument samples provided by the *Freepats*[9] project.

[6] Music Information Retrieval Evaluation eXchange
[7] http://timidity.sourceforge.net
[8] http://www.gnu.org/licenses/gpl-3.0.html
[9] http://freepats.zenvoid.org

Vienna Symphonic Library

The *Vienna Symphonic Library*[10] (VSL) is a Vienna based commercial vendor of high quality instrument samples. The provided samples do not only cover a wide range of musical instruments but also an extensive repertory of different playing styles on the individual instruments. In addition, a proprietary sequencer plug-in analyses the stream of MIDI events, detects repeated notes or other transition patterns, and either relies on according samples of that very transition or adapts the articulation in real-time. An example, where note transitions are sampled as well in order to yield more natural sounds, are legato passages on wind or string instruments.

The VSL product version used here was a *Special Edition – Standard* including the Bösendorfer 290 "Imperial" – i.e., the same type of grand piano as described above featuring the SE system. This is of special value, since it allows for an in-depth comparison between original audio recordings and audio material generated using the corresponding instrument samples. However, the fact that the VSL also offers an additional *Vienna Imperial* library, comprising a sampling of the Bösendorfer 290 with up to 1200 samples per key (including different pedal positions, note duration dependent decays, and different microphone positions), suggests that using the *Special Edition – Standard* for the rendering will result in slight deviations from the sound of the real instrument. From the reduced number of available samples, one can conclude that a number of acoustic nuances are generated by interpolation.

The VSL is a sample library including special purpose software to select the most appropriate instrument samples in real-time during playback. However, it is not a MIDI sequencer software on its own. Therefore, *Garage Band*[11] was used here.

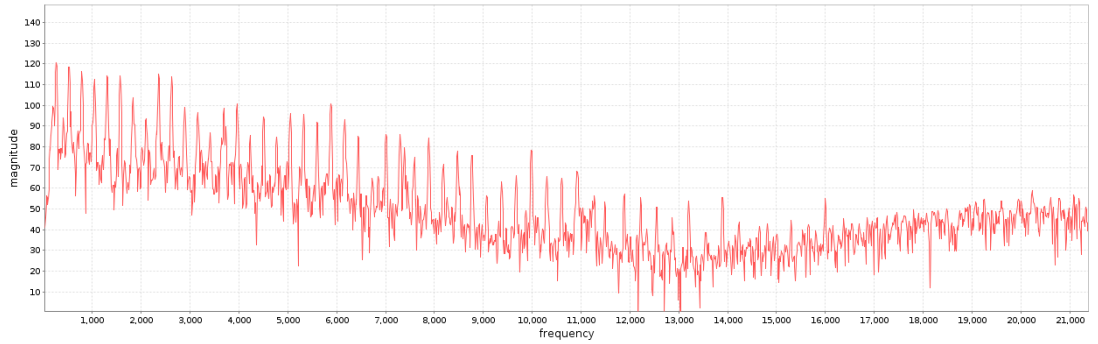
Influence on Audio-to-Score Alignment Evaluation

In [Niedermayer et al., 2011a] the influence of these different rendering methods on the results of Audio-to-Score Alignment evaluation has been studied. Audio recordings of the first movements of the Mozart sonatas were generated using the described synthesizers – timidity and the VSL. Alignments were calculated according to the state-of-the-art approach based upon Chroma vectors and Dynamic Time Warping as described in Chapter 3 and Chapter 4, respectively. An aligned note onset was considered to be correct if its absolute time displacement with respect to the actual onset time was less than 50 ms.

[10] http://vsl.co.at
[11] http://www.apple.com/ilife/garageband

(a)

(b)



(c)

Figure 2.3.: Spectra of a C3 as played on the Bösendorfer grand piano (a) and synthesized by timidity (b) and the VSL (c) calculated applying a Blackman-Harris window of length 8192 starting 50 ms after the note onset

We could observe that the ”real” audio from the SE 290 (cf. Section 2.1.2) yielded the best results (see Table 2.3 right-hand side). Examination of the spectra of individual notes generated by all three rendering methods (see Figure 2.3) revealed an increasing noise level in the higher frequency bins of the sound obtained from timidity. Since the used Chroma feature is calculated on the entire spectrum of a frame, this noise is carried over to the feature values. This phenomenon was observed to be consistent throughout the whole pitch range and is therefore a likely explanation why the system was not able to benefit from the allegedly low complexity of the audio material. When comparing

the spectra of individual notes generated by the VSL, a deviation, with respect to the recordings from the Bösendorfer grand piano, concerning the weights of the tones' harmonics (cf. the spectral peaks shown in Figure 2.3) can also be observed.

To obtain a deeper insight into the significance of these findings, a second algorithm was evaluated using the same three data sets. Onset Detection was chosen as a representative task, which also has an impact on Beat Tracking, Tempo Estimation, or Audio Transcription. As the specific algorithm the one which ranked 1^{st} in the MIREX 2010 evaluation campaign was chosen. It is based on a neural network classifier computing a pseudo-probability of an onset for each frame. The actual onsets are then determined by thresholding and peak picking (see [Eyben et al., 2010] for details).

The performance of the algorithm is determined analogously to the MIREX Onset Detection evaluation. The reported onsets are compared to the ground truth allowing a timing deviation of ± 50 ms. The quality of the result is then given in terms of the f-measure. It should be remarked that the evaluation presented here deviates from the one performed at MIREX in one aspect, which is, however, justified by the nature of the ground truth data. Merged onsets, i.e. two adjacent onsets which are reported as one single onset, are not penalized here. Since onset times are considered at the note-level instead of the chord-level, it occurs that there is more than one onset within a single or two adjacent audio frames. Such onsets cannot be distinguished without also transcribing the notes' pitches.

The results of the experiments are shown in Table 2.3 (left-hand side). Here, the performance on the data synthesized using the Vienna Symphonic Library is the highest on all individual pieces with only one exception – k.283-1 – where the signal from the SE 290 yields the highest f-value. On the other hand, the audio data obtained from timidity results in the lowest f-measure for each piece, which is consistent with the observations made in the audio alignment task. The confirmation of this effect contradicts the possible speculation that lower quality synthesizers (instrument patches) would produce somehow "artificial" sounds and in doing so reduce the complexity of the resulting audio file. On the contrary, the experiments show a very strong tendency of low quality audio samples (at least those used by timidity) to produce lower accuracy values in an algorithm evaluation.

However, when comparing the results obtained from the VSL data and the recordings from the Bösendorfer SE 290 respectively, there is no such general trend. While the onset detector performs better on the VSL dataset, the accuracy of the alignment is higher on the Bösendorfer data. We, therefore, conclude that it cannot be determined from the data source, if an evaluation data set is generally "hard" or "easy". Instead,

piece	# notes	Onset Detection			Audio-Score Alignment		
		SE 290	**VSL**	**timidity**	**SE 290**	**VSL**	**timidity**
k.279-1	2803	96.31	98.00	92.11	90.37	85.52	87.73
k.280-1	2491	98.08	98.80	95.64	85.27	79.37	85.47
k.281-1	2648	95.83	97.83	92.20	88.37	85.08	86.48
k.282-1	1907	97.70	98.87	96.42	76.68	71.93	74.93
k.283-1	3304	97.08	96.53	92.45	93.89	85.05	90.89
k.284-1	3700	94.82	98.58	93.40	92.08	90.35	86.97
k.330-1	3160	97.19	99.32	95.50	95.13	90.03	90.19
k.331-1	6123	98.02	98.50	95.55	73.00	66.62	70.70
k.332-1	3470	94.84	98.26	94.01	87.61	83.52	81.07
k.333-1	3774	96.83	98.31	93.13	93.51	93.19	92.29
k.457-1	2993	95.92	96.80	92.33	88.31	79.45	80.09
k.475-1	1284	96.69	98.29	95.60	61.21	59.04	43.04
k.533-1	4339	95.30	98.11	94.06	92.90	87.14	89.91
all	41994	96.51	98.18	94.00	86.85	81.93	82.99

Table 2.3.: Performance of a simple Audio-to-Score Alignment and an onset detector on the datasets generated using different rendering methods

this depends on the specific behavior or the fine tuning of a respective algorithm. The summarized results are shown in Table 2.3.

2.4.2. Performance Aspects

For the comparison of different audio generation methods, the original performance records in MIDI format including all expressive variations were used. However, in practice such rich data is not always available. Therefore, the influence of individual stylistic elements – usage of the pedals, changing dynamics, and micro-timing, i.e., arpeggiations and asynchronies – are also an issue.

Pedal Usage

The SE system monitors the exact pressure the pianist puts on the pedals of the grand piano. To get an idea of the influence of pedal usage on the performance of the state-of-the-art alignment algorithm, audio material was generated while neglecting all pedal information. The chosen synthesizer was timidity, due to its application in numerous research activities. Also, it must be expected that the VSL's proprietary software, choosing different rendering parameters depending on the musical context, would interfere with the experiment.

In [Niedermayer et al., 2011a] no significant differences of the respective evaluation results including or neglecting pedal usage could be reported neither for the onset detector nor for the alignment. A possible explanation is that, in contrast to music of the romantic era, such as the piano works of Frédéric Chopin, the pedal plays a minor role in the Mozart sonatas, especially during the first movements due to their general style. Table 2.4 shows the absolute and relative time the sustain and the soft pedal were pressed during the individual movements. A pedal is assumed to be pressed if the according sensor value exceeds 40, i.e., about 15% of the maximum. This threshold might seem to be relatively low. However, a clustering of all values reveals a good class separation around this value. The sostenuto pedal was disregarded due to its minor musical importance. While the usage of the soft pedal is relatively consistent amongst the individual movements, the amount of time the sustain pedal is pressed is significantly higher for the second movements.

pedal	**1^{st} movement**		**2^{nd} movement**		**3^{rd} movement**		**all**	
	time	**rel**	**time**	**rel**	**time**	**rel**	**time**	**rel**
Sustain	32:26	35.9%	44:53	57.7%	26:13	35.6%	103:32	42.8%
Soft	34:05	37.7%	29:00	37.3%	29:13	39.6%	92:18	38.2%

Table 2.4.: Absolute and relative time the pedal was pressed for the individual movements

Dynamic Variations

Dynamics is an important means of expression and a crucial element of a natural performance. In contrast to its contribution to the listening experience, it may impede MIR algorithms due to the resulting differences in absolute feature values. Although most state-of-the-art approaches to music signal processing apply some kind of normalization, varying dynamics remains a problem. While loud and soft passages can be equalized using means such as moving averages, different loudnesses of notes played approximately simultaneously are hard to resolve from the spectrogram. Dominant notes can mask soft notes, especially when they share a considerable amount of their harmonics.

Table 2.5 shows that there is a relatively consistent mean MIDI velocity deviation of about 30% for degrees of polyphony greater than two. An in-depth analysis reveals that there is a significant tendency towards larger average velocity deviations in the second movements (see Tables A.4 – A.6 in Appendix A.2 for details).

However, in analogy to the pedal pressure, dynamic variations turned out to have only insignificant influence on the evaluation results in our experiments. Even in conjunction

	all movements				
poly.	count	min	mean	(stddev)	max
1	37737	–	–	–	–
2	16206	0.000	22.823	(14.593)	94.203
3	6782	0.000	29.295	(13.814)	97.917
4	2047	1.887	32.688	(14.665)	94.915
5	452	0.000	30.776	(12.126)	85.714
6	127	8.696	32.761	(10.667)	63.415
7	22	22.472	41.444	(14.109)	91.919
8	32	14.286	29.539	(7.880)	50.000

Table 2.5.: Intra-chord dynamics deviations according to the degree of polyphony (poly.), measured as the relative difference of the softest note's MIDI velocity (m) and the loudest note's MIDI velocity (M) (calculated as $100\,\frac{M-m}{M}$)

with suppressing the usage of the pedals, deliberately "cleaning" the symbolic performance record from this expressive device by setting the velocity value of each note to a constant, did not interfere with the performance of the audio alignment as well as the onset detection.

Micro-Timings

The third expressive degree of freedom examined here are micro-timings. Human pianists will inherently not be able to play polyphonic pieces without any asynchronies. On the one hand, it is a well known phenomenon that melody notes – or in general, notes representing an emphasized voice – are played louder and, by trend, also about 30 ms earlier. [Goebl, 2001] shows that this melody lead is a direct consequence of the notes being played with different loudness, i.e., the pianist's fingers hit the keys at the same time, but due to different key and hammer velocities the strings are struck at different points in time.

On the other hand, stylistic elements such as arpeggiations or grace notes loosen up individual notes' timing constraints. While at least one of the notes which are simultaneous to each other in the score is played at its dedicated time to keep the tempo and rhythm, others are deliberately early or delayed. In Table 2.6 and Table 2.7 the magnitude of these chord spreads is illustrated. Even when disregarding ornamentation notes, such temporal spreads regularly account for several hundredth of a second. When ornamentation notes which do not have a dedicated timing but are related to a chord are included, those chords spread over up to one second.

Again, this expressive detail was removed from the symbolic performance record. To this end, the notes of each chord were given a uniform onset time, determined as the

	all movements				
poly.	count	min	mean	(stddev)	max
1	38243	–	–	–	–
2	16171	0.000	0.016	(0.020)	0.447
3	6563	0.000	0.020	(0.021)	0.514
4	1899	0.000	0.030	(0.038)	0.435
5	315	0.002	0.051	(0.059)	0.330
6	63	0.005	0.081	(0.068)	0.215
7	7	0.010	0.023	(0.013)	0.051
8	16	0.006	0.137	(0.144)	0.366

Table 2.6.: Time spreads (in seconds) between the earliest and the latest note of a chord (even if it has a notated arpeggio) according to the respective degree of polyphony (poly.), disregarding ornamentations, i.e., grace notes and trills

	all movements				
poly.	count	min	mean	(stddev)	max
1	103	–	–	–	–
2	532	0.000	0.033	(0.042)	0.248
3	369	0.001	0.070	(0.080)	0.478
4	196	0.007	0.139	(0.108)	0.691
5	148	0.014	0.243	(0.133)	0.637
6	65	0.090	0.227	(0.119)	0.621
7	15	0.090	0.337	(0.156)	0.598
8	16	0.092	0.194	(0.210)	1.001

Table 2.7.: Time spreads (in seconds) between the earliest and the latest note of a chord including ornamentations, where auxiliary notes without a dedicated timing in the score (e.g., trills, gracenotes, mordents) are compared to the note their are ornamenting and auxiliary notes with a notated timing are compared to other notes having the same score time

mean over all individual onsets. In contrast to pedal usage and dynamic variations, asynchronies exhibit a significant influence on MIR evaluation results. For Audio-to-Score Alignment, a major improvement was achieved at each individual piece. Especially in those examples where the alignment on the original data performs worst, the unified onsets caused remarkable improvements of alignment accuracy. For the piece k.475-1, the percentage of well aligned onsets almost doubled from 43% to 81% (see Table 2.8).

In contrast, the onset detector could not improve on each piece but, for some examples, underperformed on the modified data. An in-detail inspection showed that the unified onsets cause such exceptionally high peaks within the detection function, that onsets of notes played one at a time are likely to be masked out.

piece	Onset Detection		Audio Alignment	
	full	**time**	**full**	**time**
k.279-1	92.11	98.10	87.73	95.33
k.280-1	95.64	99.30	85.47	95.19
k.281-1	92.20	82.53	86.48	91.66
k.282-1	96.42	92.55	74.93	96.89
k.283-1	92.45	97.15	90.89	99.64
k.284-1	93.40	99.52	86.97	98.57
k.330-1	95.50	89.56	90.19	96.52
k.331-1	95.55	98.49	70.70	99.11
k.332-1	94.01	99.15	81.07	99.17
k.333-1	93.13	99.73	92.29	96.88
k.457-1	92.33	99.32	80.09	95.07
k.475-1	95.60	91.56	43.04	80.58
k.533-1	94.06	92.24	89.91	97.29
all	96.51	96.01	82.99	96.61

Table 2.8.: Performance of the example algorithms on the datasets exhibiting all aspects of expressive variations (full) and with suppressed micro timings (time)

2.5. Conclusions and Consequences for this Thesis

From this analysis we draw two main conclusions. First, the kind of data source does not have a universal influence on the difficulty of an MIR task. Particularly, the assumption that synthesized audio material yields better results than "real" recordings is not valid. On the other hand, it became evident that audio alignment, especially when based upon standard algorithms which cannot differentiate between individual chord notes and assign a single timestamp to each chord, remarkably benefits from removed asynchronies. Despite this seeming simplification of the task, algorithms which are not designed for this type of data might fail due to the unnatural characteristics of the resulting audio material.

For this thesis, the implication of these findings is that we do not include any synthesized data in our evaluation. In principle, this would be a means of obtaining a larger corpus of audio material with a corresponding ground truth transcription. However, we lack additional performance MIDI files, i.e., transcriptions of performances by skilled pianists, and have, therefore, decided to rely on the Mozart corpus as described above instead of generating artificial performances.

Also, including synthesized versions of the performances of the Mozart sonatas into the evaluation set is unlikely to reveal new insights. As shown in Table 2.3, piece and performance have a greater influence on the accuracy of an alignment result than the audio source. Again, we decided to rely on the data corpus as described above.

3. Feature Extraction for Audio Alignment

In this chapter, the extraction of meaningful descriptors from audio signals as well as from the MIDI representation of the score is described. Since the latter is in a symbolic format, this step can basically be reduced to simple mappings of this high-level representation into the respective feature domain. The audio recording, on the other hand, is a digitized representation of sound waves and – analogous to a printout of a plotted waveform – as such not intuitive. Timbre, pitch, dynamics, and note timings as well as durations are high level concepts which are unconsciously processed by the human auditory and perceptive system but, in contrast, not obvious to computer systems.

Several MIR fields, such as Onset Detection, Multi-Pitch Estimation, or Audio Transcription, concern themselves with the extraction of high level descriptions from audio signals. While some approaches yield good results in constrained environments, they are in general not robust enough to describe arbitrary audio signals. In the MIREX 2010[1] algorithm evaluation the highest f-measures were below 0.8 for Onset Detection and below 0.6 for piano-only Multi-f_0 Tracking, where a note was assumed correct if the pitch was correct and its onset time was allowed to be within a tolerance range of ± 50 ms centered around the real onset. The same tolerance of ± 50 ms was applied for the evaluation of the onset detectors.

What makes these tasks difficult, is to decide if evidence for an onset or a pitch is significant or results from noise or interference patterns. The decision making is generally implemented as a peak picking routine or by classification, based on machine learning techniques. Both approaches, however, are often fragile and prone to errors. Therefore, most audio alignment systems prefer features which can be calculated without such a decision process and accordingly do not require any training or parameter tuning step.

[1] http://www.music-ir.org/mirex/wiki/2010:MIREX_HOME

In this chapter, three features are presented. Two of them – Chroma vectors and onset based features – are generally known and used in the initial alignment step of our proposed Audio-to-Score Alignment system (see Chapter 6). The system's refinement stage uses the third feature – Pitch Activation. We initially proposed this feature for audio alignment in [Niedermayer, 2009a] and have, then, further investigated into adaptations and improvements.

3.1. Time-Frequency Transformations

The vast majority of audio features relevant for Audio-to-Score Alignment – in particular all variants of the *Chroma vectors* described later in this section – are extracted from the frequency domain representation of a signal. Since temporal information is crucial to sound and music computing, for example, a Cosine transform of the entire signal is not sufficient. Instead, a two-dimensional time-frequency representation is needed. This section gives an overview of the corresponding methods applied throughout this work and described in the Audio-to-Score Alignment literature.

3.1.1. Short Time Fourier Transform

The *Short Time Fourier Transform* (STFT) is the time-frequency transform most commonly used in the MIR literature. It is based on the *Discrete Fourier Transform* (DFT) of a discrete periodic signal x_n with a period of length N defined as

$$DFT(x) := X_k = \sum_{n=0}^{N-1} x_n e^{-i\frac{2\pi kn}{N}}, \tag{3.1}$$

where $k \in [0, N)$. The vectors $e^{i\frac{2\pi kn}{N}}$ form an orthogonal basis with resulting coefficients X_k being complex numbers. According to Euler's formula they represent sinusoidal components of amplitude

$$A_k = \sqrt{\Re(X_k)^2 + \Im(X_k)^2} \tag{3.2}$$

and phase

$$\varphi_k = \arctan\left(\frac{\Im(X_k)}{\Re(X_k)}\right) \tag{3.3}$$

The calculation of the DFT according to Equation 3.1 has an asymptotic complexity of $O(n^2)$. A computationally efficient method to obtain the DFT is the *Fast Fourier Transform* (FFT) which reduces the complexity to $O(N \log N)$. Although the idea of the FFT dates back to *Carl Friedrich Gauss* (cf. [Heideman et al., 1984]), the first algorithm was proposed in [Cooley and Tukey, 1965]. It requires that the length of the signal's period can be factorized into small primes – ideally it is a power of 2. The so called radix-2 FFT then recursively splits the input signal into a sequence of even-indexed samples and a remainder sequence of odd-indexed samples. After calculating the DFT on the subsequences, the individual results are combined in order to obtain the transform of the whole signal. The processing follows a tree, where the root is the entire signal and at each node the remainder signal is split into two. After the calculation of the DFT at the leaves, which are sequences consisting of two samples each, the tree is traversed postorder, joining the results at each node. A method for rearranging the samples of the signal, such that the order resembles that of the leaves, is to move each sample to its bit-wise reversed index. This variant is called a *decimation-in-time* FFT, in contrast to *decimation-in-frequency* FFT algorithms, where the transform is performed first and then a rearrangement takes place. Throughout this work, the *FFTW* library ([Frigo and Johnson, 2005]), which offers several variants of the FFT computation which are applied depending on the problem size as well as on the machine, is used the obtain the DFT of a signal.

To obtain a time-frequency representation an additional mechanism is needed. The DFT yields a spectrum of the input signal but no time information, i.e., no differentiation if a frequency is present during the whole signal or only at certain sections. Therefore, the signal is split into short fragments which are assumed to be approximately stationary and a spectrum is calculated for each of these frames. The resulting 2-dimensional representation is called *spectrogram*. For a large number of applications the log-power variant of the spectrogram, based on

$$\hat{A}_k = 20 \log A_k \tag{3.4}$$

is utilized in accordance with the human auditory system.

Calculating the DFT on one signal chunk of length M at the time can be rephrased as

$$STFT(x)_w := X_{k,w}(\tau) = \sum_{n=-\infty}^{\infty} x_n w_{n-\tau} e^{-i\frac{2\pi kn}{N}} \tag{3.5}$$

where

$$w_n = \begin{cases} 1 & \text{if } n \leq \frac{M-1}{2} \\ 0 & \text{else} \end{cases} \tag{3.6}$$

and $\tau = t\,M_{hop}$ is an integer multiple of the hop size M_{hop}.

The used window function represents a rectangular window, which is a simple, nevertheless unfavorable option. The transient cutoffs at the window boundaries will most likely conflict with the periodicity assumption of the DFT. Also, inspecting the window in the frequency domain, where the convolution in Equation 3.5 becomes a multiplication, shows that the frequency response of the window has non-zero values not only at the main lobe centered around the corresponding frequency but also at side lobes. Those side lobes are responsible for the duplication of spurious energy content from adjoining frequency bins – an effect which is called *spectral leakage*. This leakage effect is inherent to the STFT. However, different window functions differ in how fast the magnitudes of the side lobes decay.

Common choices for window functions are the *Hamming* window, defined as

$$w_n := 0.54 - 0.46\,\cos\left(\frac{2\pi n}{M-1}\right) \tag{3.7}$$

and the *Hann* window, defined as

$$w_n := 0.5\left(1 - \cos\left(\frac{2\pi n}{M-1}\right)\right) \tag{3.8}$$

While the Hamming window has a clearer separation between the main lobe and the first side lobe of more than -40 dB, the Hann window shows a faster general decay of the side lobes' magnitudes of about -18 dB per octave. For a description of other common

window functions, the interested reader is referred to the signal processing literature, such as [Oppenheim and Schafer, 2007].

A second trade-off, besides the choice of the window function, is the adequate selection of the window size M. While a large window yields a good frequency resolution, small windows give the advantage of a higher time resolution. In the literature, common window sizes vary between 512 and 8192 samples. At a sampling frequency of 44.1 kHz this results in windows lengths of 11.6 ms up to 185.8 ms. The corresponding frequency resolutions, i.e., the differences between two adjacent bins' center frequencies, are 43.1 Hz and 2.7 Hz respectively. In comparison, the two lowest tones of standard grand pianos – the A0 and the A♯0 – have a frequency difference of 1.636 Hz only. Accordingly, the window of size 8192 has a fine enough frequency resolution to distinguish pitches from F♯1 upwards. Frequency differences of around 40 Hz cannot be found between pitches lower than F4.

Two methods which seemingly increase the attainable time-frequency resolution are zero-padding and overlapping windows. Padding zeros at the begin and the end of each windowed signal chunk yields an STFT result having a higher number of frequency bins. However, the relating coefficients do not contain additional information, but are "blurred" such as when interpolating between a limited number of values.

Overlapping windows, on the other hand, are a necessary means to avoid data loss. Since common window functions, except from the rectangular window, converge towards zero or very small values as the index approaches the boundary, the signal samples multiplied by those resulting values are almost canceled out. To avoid this effect, overlap ratios of 50% or more are beneficial. However, decreasing the hop size beyond a certain threshold will not yield more information but, again, result in an interpolation-based oversampling.

There is no principled way of finding an optimal window size. A well known approach to circumvent this decision is to use several spectrograms calculated at different time-frequency resolutions in parallel (see [Eyben et al., 2010], for example). A similar method is to use different window lengths depending on the frequency bands under consideration. This approach leads to the idea behind the constant Q transform, which will be discussed later.

In applications where only the peak frequencies are of interest, the effective frequency resolution can be further improved independent from the actual time-frequency transform. In [Gómez, 2006] a quadratic spectral interpolation based on the peak value, the values of its two adjacent bins, and the assumption that those three points lie on a parabola is performed. Alternatively, the phase information contained in the

STFT coefficients can be used to calculate the instantaneous frequency, as described in [Flanagan and Golden, 1966], [Boashash, 1992], or [Goto and Hayamizu, 1999]. The complex coefficient $X_k(\tau)$ is rearranged as $X_k(\tau) = a_k(\tau) + i\, b_k(\tau)$. Then the instantaneous frequency $\hat{f}_k$ is calculated as

$$\hat{f}_k(\tau) = f_k + \frac{f_s}{2\pi M_{hop}} \frac{b(\tau)(a(\tau + M_{hop}) - a(\tau)) - a(\tau)(b(\tau + M_{hop}) - b(\tau))}{a(\tau)^2 + b(\tau)^2} \tag{3.9}$$

or when considering an individual frame, as

$$\hat{f}_k = f_k + \frac{f_s}{2\pi M_{hop}} \frac{b\Delta a - a\Delta b}{a^2 + b^2} \tag{3.10}$$

where f_k is the kth bin's center frequency and f_s is the sampling frequency.

3.1.2. Constant Q Transform

An inherent property of the Short Time Fourier Transform and the Discrete Fourier Transform in general, is that the frequency bins are linearly spaced. However, this linear partitioning of the frequency scale does not correspond to the human auditory system and the way humans perceive pitch. Considering the standard pitch range of a piano, the frequency difference between its two lowest tones (MIDI pitches 21 and 22) is 1.635 Hz. It then increases by a factor of $2^{\frac{1}{12}}$ for each consecutive interval up to a frequency difference of 234.9 Hz (between the MIDI pitches 107 and 108). Applying a DFT is problematic, since only very long windows are able to yield a frequency resolution high enough to discriminate individual pitches at the low frequency bands. This high resolution is maintained at the higher, sparse frequency bands, however, at the cost of a low time resolution.

A time-frequency transform more adapted to the human pitch perception is the *Constant Q Transform* (CQT, see [Brown and Puckette, 1992], [Schörkhuber and Klapuri, 2010], or [Velasco et al., 2011], for example). The quotient Q represents the relation between a bin's center frequency f_k and its bandwidth

Δf as $Q = f_k/\Delta f$. A constant value of Q implies that the center frequencies are geometrically spaced. The transform is defined as

$$CQT(x) := X_k = \sum_{n=0}^{N-1} x_n e^{-i\frac{2\pi\, n\, f_k}{f_s}} \tag{3.11}$$

where

$$f_k = f_{min}\, 2^{\frac{k}{12a}} \tag{3.12}$$

and a is the number of frequency bins per semitone on the equal tempered scale. The number of frequency bands must be chosen in such a way that the highest resulting frequency $f_{k_{max}}$ is below the Nyquist frequency.

[Brown and Puckette, 1992] propose a computationally efficient algorithm for the calculation of the CQT. To this end, the windowing step is included into Equation 3.11 and the result is rearranged as

$$X_{k,w}(\tau) = \sum_{n=0}^{M-1} x_n a_k^*(n-\tau) \tag{3.13}$$

with

$$a(n) = \frac{1}{N_k} w(\frac{n}{N_k}) e^{-i\frac{2\pi\, n\, f_k}{f_s}} \tag{3.14}$$

and $a_k^*(n)$ being the complex conjugate of the atoms $a_k(n)$. The size N_k of the respective windows is inverse to the center frequency f_k and adapted for each bin, such that the quotient Q remains constant. The window function $w(n)$ is supported in the interval $[0, 1]$. According to Parseval's theorem,

$$\sum_{n=0}^{N-1} x_n a_k^*(n) = \sum_{m=0}^{N-1} X(m) A_k^*(m) \tag{3.15}$$

where $X(m)$ and $A_k(m)$ denote the DFTs of $x(n)$ and $a_k(n)$ respectively. The spectral kernels A_k^* can be precalculated and stored, reducing the computation of the CQT to

performing an FFT and a matrix multiplication according to Equation 3.15 for each frame. The matrix multiplications are computationally cheap due to the sparseness of the kernels.

An inverse CQT and additional performance enhancements were proposed by [Schörkhuber and Klapuri, 2010]. By processing one octave at a time, considerable differences in the lengths of the required windows can be avoided. This modification does not only improve the sparseness of the kernels but also allows for longer hop sizes in the lower frequency regions. An alternative computation of the inverse constant Q transform is presented in [Velasco et al., 2011] based on non-stationary Gabor frames.

3.1.3. Wavelet Transform

An operation similar to the Constant Q Transform is the *Discrete Wavelet Transform* (DWT). As in the Fourier analysis, the signal is projected linearly on a function base. However, in contrast to the sine and cosine functions which are supported on $\mathbb{R}$, wavelets, i.e., the basis functions of the DWT, exhibit only short ranges in which their values are different from zero. Individual wavelets are constructed based on either a *mother wavelet* $\psi(t)$ or a *scaling function* $\phi(t)$. The simplest such wavelet is the *Haar wavelet* defined as

$$\psi_{00}(t) := f(t) = \begin{cases} 1 & 0 \leq t < \frac{1}{2} \\ -1 & \frac{1}{2} \leq t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

The wavelets most commonly used in the literature are the ones of the *Daubechies family* (see [Daubechies, 1988]) which are recursively constructed from the Haar wavelet. They were proven to have a number of beneficial properties such as orthogonality and the highest possible number of vanishing moments. The actual DWT is then defined as

$$DWT(x)_\psi := X_{k,\psi} = \sum_j \sum_n x_n 2^{-j/2} \psi(2^{-j}k - n) \tag{3.17}$$

where the mother wavelet $\psi(t)$ is only contracted and dilated by powers of 2.

[Mallat, 1989] introduces a computationally efficient algorithm of an asymptotic complexity of $O(n)$ to compute the DWT of a signal. To this end, a pair of *quadrature mirror filters* corresponding to a certain wavelet are constructed such that one is a low-pass whereas the other one is a high-pass filter. In each iteration of the algorithm, the two respective filters are applied to the remainder signal and according outputs are downsampled by a factor of 2. The high frequency part of the signal is kept as the detail information at the level corresponding to the current iteration. The low frequency part is further decomposed using the next pair of filters. In doing so, at each step the frequency resolution is doubled – because the frequency range of the input signal is divided into a high frequency and a low frequency part – while the time resolution is set to the half – due to the downsampling. Therefore, the result is similar to the outcome of a CQT (cf. [Bayram and Selesnick, 2009], [Brown and Puckette, 1992], or [Tzanetakis et al., 2001]).

3.1.4. Gabor Analysis

The *Gabor analysis* can be seen as a generalization of the STFT. Instead of dividing the time-frequency space into a fixed grid according to the window and hop sizes applied in an STFT, Gabor analysis is based on time-frequency atoms which are derived from a single prototype $g(t)$ by translation, i.e., time shifts, and modulation, i.e., frequency shifts. Given these two operators, defined as

$$T_k\, f(t) := f(t+k) \tag{3.18}$$

and

$$M_l\, f(t) := e^{-i\frac{2\pi\, lt}{L}} \tag{3.19}$$

respectively, the signal x can be expressed as a linear combination of multiple instances of the atom $g(t)$ shifted in the time-frequency space

$$x_n = \sum_{t,k\in\mathbb{Z}} X_{t,k} M_{kb} T_{ta} g(n) \tag{3.20}$$

where a represents the width of a time shift and b the width of a frequency shift. The atom $g(t)$ is a function, such as the Gaussian. It is the equivalent to the window

function applied during the computation of an STFT. To compute the coefficients $X_{t,k}$, the set of shifted atoms must form a frame, i.e., a basis, however, without the linear independence constraint. In addition an inner product $\langle ., . \rangle$ must be defined on the time-frequency space. Then there exists a corresponding dual frame $M_{bk}\, T_{at}\, \gamma$, such that

$$X_{t,k} = (\langle x, M_{bk}\, T_{at}\, \gamma \rangle)_{t,k} \tag{3.21}$$

While shifting the atom along a fixed grid according to the parameters a and b does not offer much advantage in comparison to the STFT, the concept of *Multiple Gabor Frames* does. [Dörfler, 2002] describes the idea to have not only a single atom, but various prototypical functions according to the expected signal content at certain regions of the time-frequency space. At times where note onsets are expected, frequency resolution is not critical, especially for instruments where a note onset causes a transient energy burst throughout the whole spectral range. In such cases an atom aiming at accurate time resolution can be applied. On the other hand, during the sustain phase of a note, atoms designed for higher frequency resolution can be applied.

However, as for all time-frequency transforms, the capabilities of Gabor analysis in terms of time-frequency resolution are limited by Heisenberg's uncertainty principle. Placing the atoms too dense within the time-frequency space, results in the same effects as can be observed when zero-padding or highly overlapping windows are applied during an STFT computation [Dörfler, 2002][Dörfler, 2004].

3.1.5. Filter Banks

While the above described methods are direct transforms of a signal into the frequency domain, digital audio filters are no such transform per-se. However, an adjusted bank of bandpass filters can separate the components of a signal in a fashion similar to Fourier analysis. On the other hand, designing a filter bank offers more flexibility than the application of, for example, an STFT. When, instead of bandpass filters, comb filters are used, the energy contribution of a certain pitch including not only its fundamental frequency but also its harmonics can be obtained in a single transform step.

In [Müller et al., 2004] a bank of 88 bandpass filters corresponding to the fundamental frequencies of the notes available on a standard piano is applied. The filters are implemented as 8^{th}-order elliptic filters having a rejection of 50 dB in the stop-band. To keep the computational costs low, the signal is downsampled such that the high,

medium, and low frequency content is computed based on three versions of the signal at a respective sampling rate.

A time-frequency representation is obtained from the filter output by calculating the short-time root-mean-square power (STRMS power) for each of the frequency bands. Depending on the sampling frequency at which a certain filter output was computed, a window function with an according interval of support is applied.

Other commonly applied filterbanks are based on auditory models, such as proposed by [Patterson et al., 1992], simulating the human perception of sound, i.e., the behavior of the cochlea and the basilar membrane. A basic concept is the partition of the audible frequency range into 24 critical bands according to the *Bark* scale. Frequencies within such a band are subject to a conjoined analysis which can result in masking effects. The actual filters are commonly implemented as 4^{th}-order gammatone filters due to their cutoff characteristics and computational efficiency (cf. [Painter and Spanias, 2000], for example).

Another approach is described by [Cheveigné, 1993], reviewing a method for the detection of pitches present in a signal by tuning comb filters to narrow pass bands at the fundamental and harmonic frequencies of a pitch. This is similar to the dictionary based spectral factorization described in Section 3.4, however instead of performing a time-frequency transform first and then decomposing the signal, comb filters are applied directly on the time domain signal.

3.1.6. Discussion

In the sections above, some of the most commonly applied time-frequency transforms have been briefly described. While for some of them, such as the STFT and the CQT, the inherent differences are apparent, others, such as filter banks, are flexible instruments that can be used interchangeably with other approaches. In such cases, considerations concerning computational efficiency or appropriateness of data representations become determining.

In numerous preliminary experiments, we found the short-time Fourier transform with a relatively high window overlap ratio to be best suited for most algorithms described within this thesis. One might argue that it is not adequate for the processing of musical audio due to its linear spacing of frequency bands, which disregards the logarithmic frequency spacing of pitches as perceived by human listeners. While this reasoning is sound in principle, it neglects the fact that a pitch is not only represented by its fundamental frequency but also by its harmonics. Not considering the small inhar-

monic deviations characterizing a certain instrument, harmonic frequencies are integer multiples of the fundamental frequency. While the interval between the fundamental and the first harmonic frequency is an octave, the same absolute frequency difference equals less than a semitone between the 16^{th} and the 17^{th} harmonic. While most systems described in the literature do not consider harmonics of such a high order, this figure, nevertheless, demonstrates that partials of a signal become more compact on a pitch scale in higher frequency ranges. While the geometrical spacing of frequency bins of the CQT accommodates the fundamental frequencies of notes it disregards the harmonics and the discrimination between partials of different tones. This also holds for the Discrete Wavelet transform.

A second argument is computational efficiency. To not lose data at the high frequency bins of the CQT, where the window sizes are relatively short, the hop size must be accordingly small. However, this results in highly overlapping windows at the low frequencies, which require very long windows which are computationally expensive. When using an STFT one can obtain an additional spectrogram aimed at the recognition of low notes in an efficient way from an accordingly downsampled version of the input signal. Nevertheless, we use the CQT in certain refinement steps of our proposed system, where we are not interested in the tonality of a frame but in the exact energy at the fundamental frequencies of individual pitches.

Comparing the STFT to the more general multiple Gabor frames, the STFT is more efficient again. While the multiple Gabor frames approach is a self-contained, orderly mathematical framework, from an algorithmic point of view, it is cheaper to compute several spectrograms at different constant time-frequency resolutions instead of determining adequate resolutions for each segment of the input signal. In addition, in an analysis-only setting, where the accurate resynthesis of the signal is not an issue, some of the constraints introduced by the Gabor analysis framework can easily be dropped.

Also, the equally spaced lattice of the STFT benefits computational processing due to its straightforward data structure. Gabor atoms with centers of gravity at arbitrary points within the time-frequency space are, for example, not usable for transformations by means of matrix multiplication, but would require adapted routines.

Filter banks can be implemented very efficiently. However, to gain additional benefits in comparison to the described transforms, an in-depth filter design is required. Comb filters, which are intended to take all partials of a certain tone into account, would need to be adjusted to the exact tuning and inharmonicity of a certain instrument. In accordance with numerous other authors we have decided on performing a straightforward transform and concentrating our efforts on subsequent processing step in the frequency domain.

The FFT implementation used throughout this work is the *FFTW*[2] – an open source C library developed at the MIT and distributed under the GNU GPL[3]. The FFTW contains a number of FFT algorithms and yields high performance due to automatic selection of the appropriate algorithm depending on the data and other runtime self-optimization techniques (see [Frigo and Johnson, 2005]).

3.2. Chroma Vectors

Chroma vectors, also known as *Pitch Class Profiles*, are the probably most commonly used feature for audio alignment tasks. Although the first name is used more frequently in the current literature, it was only introduced in [Bartsch and Wakefield, 2001], while the same idea had already been proposed in [Fujishima, 1999]. The Chroma feature has been used in numerous systems, not only for Audio-to-Score Alignment and Score Following, but also for solutions involving audio alignment as an intermediate step, such as in the fields of Version Detection, Structural Analysis, or Content-based Retrieval. In this very context of Content-based Retrieval, [Hu et al., 2003] presents a comparison between Chroma Vectors, Pitch Histograms and two MFCC-based approaches showing that Chroma vectors significantly outperform the other features.

Another evaluation is presented in [Joder et al., 2010a], where Chroma features obtained from different time-frequency representations are compared to spectral models and semitone energy features. The authors showed that while the spectral model based approached performed significantly worse than the others, there is no such significant difference between the results obtained using Chroma variants or features considering the energy of each individual semitone. Chroma vectors are, therefore, preferred, due to their compactness and efficiency. In addition, a comparison of different types of Chroma features is compared within the context of Chord Detection in [Jiang et al., 2011]

The basic idea behind Chroma features is to not consider a single pitch's contribution to the spectral energy of a frame, but the energy of a whole pitch class (i.e., all C, C♯/D♭, D, D♯/E♭, etc. without taking the actual octave into account). In doing so, octave errors, which account for a considerable percentage of the overall error rate in multi-pitch detection (see [Brossier, 2006], for example), are avoided. Also, by mapping all pitches into one octave, the amount of data is significantly reduced.

The probably easiest way of obtaining a chroma vector from an audio frame starts by mapping each bin's center frequency f_k into a certain octave by multiplication by the factor of 2^n, where $n \in \mathbb{N}$ is the number of octaves by which f_k is adjusted. The

[2] http://www.fftw.org
[3] http://www.gnu.org/copyleft/gpl.html

resulting frequency $\hat{f}_k$ is then compared to the fundamental frequencies of the 12 pitches within the reference octave. The k^{th} bin is then assigned to the pitch class c with its prototypical fundamental frequency nearest to $\hat{f}_k$. The class index can be calculated by

$$c(f_k) = \left[12 \log_2 \frac{f_k}{440} + 12n + 9\right] \mod 12 \tag{3.22}$$

where n is an integer factor which has to be chosen such that the term within the brackets is positive and the addition of 9 shifts the result such that the pitch class C has an index of 0.

3.2.1. Distance Weighting

Mapping a frequency bin directly to a pitch class can be problematic. It would result in each frequency bin contributing to exactly one pitch class with is entire energy independent of how accurate the folded center frequency $\hat{f}_k$ matches the nearest prototypical fundamental f_{c_i}. However, in cases where $\hat{f}_k$ lies in between two pitch class prototypes, i.e., its distance

$$d_i(\hat{f}_k) = \left|12 \log_2 \frac{\hat{f}_k}{f_{c_i}}\right| \tag{3.23}$$

to the nearest f_{c_i} in semitones is close to 0.5, the energy contribution should be divided among the respective pitch classes. From another point of view, one can argue that energy contributions of bins which cannot be accurately mapped to any pitch class should be down-weighted. To this end, a weighting function with a support interval of length l centered around a d_i value of zero is introduced. [Gómez, 2006] proposes a cosine squared weighting function defined as

$$w(d_i) = \begin{cases} \cos^2\left(\frac{\pi}{2} \frac{d_i(\hat{f}_k)}{0.5l}\right) & \text{if } d \leq 0.5l \\ 0 & \text{otherwise} \end{cases} \tag{3.24}$$

3.2.2. Spectral Peak Selection

In contrast to taking each frequency bin into account, a different method to compute the chroma vector is to only consider frequencies where the spectrum forms a peak. In doing so, one avoids energy contributions which are due to noise and concentrates on sinusoids which are supposed to be meaningful. By additionally calculating the instantaneous frequency at the spectral bins, the feature is calculated based on an accurate estimate of the partials sounding at a certain audio frame (cf. [Goto, 2005] or [Ellis et al., 2008]).

While the approach is sound, its implementation poses questions concerning an adequate peak picking. A common method is to smooth the spectrum to reduce the number of local maxima and to also apply thresholding. Strategies for the selection of the thresholds can be based on auditory principles such as the absolute threshold of hearing or masking effects, i.e., assumed partials with an amplitude too low in absolute numbers or in comparison to the amplitude of a proximate peak are dropped.

3.2.3. Harmonic Frequencies

Another enhancement proposed by [Gómez, 2006] is to also take into account that a partial found within a signal does not need to have the fundamental frequency of a note, but, with a much higher probability, might have a harmonic frequency. The direct mapping of frequencies to pitch classes, as described above, neglects this fact. Since for vibrating strings the first and the third (ideal) harmonic are of the same pitch class as the fundamental (see Table 3.1), a straightforward mapping implicitly assumes that the sum of the amplitudes of these two partials and the fundamental frequency dominates that of other partials belonging to a different pitch class. This is justified by the observation that the energy tends to decrease for each other harmonic and therefore three out of the four most dominant partials belong to the desired pitch class.

For the computation of the *Harmonic Pitch Class Profile* (HPCP), however, each spectral peak is in turn assumed to be the fundamental and the first up to the H^{th} harmonic frequency of a pitch, where $h \in [1, H]$. Then the energy of the i^{th} pitch class is defined as

$$HPCP(i) = \sum_{f} \sum_{h=0}^{H} w\left(d_i\left(\frac{f}{h+1}\right)\right) A(f) s^h \tag{3.25}$$

harmonic	**interval**	**PC offset**
fundamental	(0)	(0)
1	12	0
2	19.0	-5.0
3	24	0
4	27.9	3.9
5	31.0	-5.0
6	33.7	-2.3
7	36	0
8	38.0	2.0
9	39.9	3.9
10	41.5	5.5

Table 3.1.: Interval in semitones between the fundamental frequency and the harmonics and the respective offsets in terms of pitch classes

where $d_i(f)$ and $w(d_i)$ are distance and weighting functions according to Equation 3.23 and 3.24 respectively, $A(f)$ is the amplitude, and $s \in (0, 1]$ yields a constant or a decaying weight for each consecutive harmonic.

3.2.4. Pre- and Post-processing Methods

In order to enhance the explanatory power of the feature, several authors have proposed additional pre-processing methods. One of the probably simplest approaches is to introduce a silence detection as, for example, done in [Niedermayer, 2008]. Numerous audio features, including Chroma vectors computed without a normalization, can represent silence in the feature domain. Normalization, however, discards all information on the absolute loudness or the absolute energy within an audio frame. To maintain this piece of information, a detection of silent passages can be performed on the amplitude envelope of the time domain signal or on the total energy of a frame in the frequency or feature domain by simple thresholding.

A second method for pre-processing audio signals is spectral whitening, as described in [Klapuri, 2003] and also used in [Niedermayer, 2008]. It aims at reducing timbral influences as well as suppressing noise. To this end, the spectrum X_t of a frame at time t is modeled as

$$X_t(k) = H_t(k)S_t(k) + N_t(k) \tag{3.26}$$

i.e., a superposition of an excited sound S_t modified by the frequency response H_t of the environment, including the instrument body or room acoustics, and a noise component N_t. The effect of H_t is lessened by a magnitude warping of X_t according to

$$Y_t(k) = \ln\left(1 + \frac{X_t(k)}{g(X_t)}\right) \tag{3.27}$$

while allowing for the noise component N_t to be linearly subtracted from the result in a second step. [Klapuri, 2003] proposes a function $g(X_t)$ defined as

$$g(X_t) = \left[\frac{1}{K}\sum_k X_t(k)^{\frac{1}{3}}\right]^3 \tag{3.28}$$

and an approximation $\bar{N}_t(k)$ calculated as the moving average over $Y_t(k)$ calculated on a logarithmic frequency scale such that the final whitened signal is

$$Z_t(k) = \max\left(0, Y_t(k) - \bar{N}_t(k)\right) \tag{3.29}$$

Other authors perform a tuning frequency estimation to obtain a more accurate mapping between frequencies and pitch classes. It is relatively common that instruments are not tuned to a standard pitch of $A4 \mathrel{\hat{=}} 440\,\text{Hz}$ but to a slightly deviating frequency. In [Gómez, 2006] an algorithm is described which extracts the instantaneous frequency of each spectral peak within the time-frequency representation of an audio recording. Building a histogram over the detuning factors obtained from these peak frequencies yields the most likely global tuning. Consideration of the exact tuning can also be found in [Dressler and Streich, 2007], [Lerch, 2006], or [Vincent et al., 2008], for example.

When two feature sequences are compared, one has to decide on an analysis window overlap ratio. On the one hand, since the common alignment algorithms have quadratic complexities, a high overlap ratio will quickly result in very high computational costs. On the other hand, features which are sparse in time bear the risk that, due to a small offset or a tempo variation, the combination of notes present during the analysis windows will not be the same when two versions of a piece are compared. To make feature sequences invariant to such offsets or tempo variations, in [Ellis et al., 2008] or [Bertin-Mahieux et al., 2010], a beat tracking is performed on the audio data. Then the Chroma feature is calculated at a fixed resolution in terms of frames per beat.

Concerning post-processing, a large number of proposed systems uses normalized Chroma vectors. Normalization is achieved by linear scaling, such that either the sum or the maximum of all values is a constant (usually 1). It should also be remarked that some authors work with resolutions finer than one semitone. This results in vector lengths larger than 12. A relatively common choice are feature resolutions of 24 or 36 elements, i.e., pitch classes (cf. [Gómez, 2006], [Sheh and Ellis, 2003], or [Müller et al., 2011]).

3.3. Onset-based Features

The feature described above – Chroma vectors – describes the tonality of a frame. Although this information allows for matching sections of the audio recording to corresponding passages within the score, this feature is not designed to yield accurate note onset estimations. To also consider exact event timings, a second class of features, which is based on onset detection functions, is introduced.

Such a feature was originally proposed in [Müller et al., 2004]. There, a multi-rate bank of 88 elliptic filters was used to obtain a time-frequency representation where each filter output corresponds to the fundamental frequency of one note within a standard piano's pitch range. The resulting short-time root-mean-square power is similar to the Pitch Activation feature, with the major difference that not all or at least several partials of a pitch are taken into consideration, but only a certain band centered around the fundamental frequency of a pitch.

Based upon this time-frequency representation, the novelty in each band is calculated by half-wave rectifying the first-order difference function over each frequency band. Local maxima are then selected as onset candidates not only for the notes with a fundamental frequency equaling a band's center frequency f_c but also for the pitches with fundamental frequencies of $f_c/2$ and $f_c/3$. In doing so, the fact that the first two harmonics are contributing a significant fraction of a note's overall energy, is accounted for. Spurious candidates are removed by applying a local threshold.

Although onset candidates are sparse and highly accurate cues for novel partials within an audio signal, their usage as features in an audio alignment task is limited. On the one hand, the onset descriptors strongly depend on the peak picking step which is error-prone due to several aspects. Shared partials, low signal novelty when notes are repeated or a passage is performed very softly, and high degrees of polyphony due to extensive usage of the sustain pedal are phenomena which hamper the detection of note onsets. On the other hand, notes of a chord are played asynchronously most of the time

during real performances. This results in note onsets at different times, contradicting the synchronous onset within the score. Due to the instantaneous character of note onsets, i.e., the fact that they do not have any temporal spread, it is unlikely that the onsets of all notes of a chord are within the same audio frame. As a consequence, the information if notes are played one at a time or as a chord is lost. Therefore, adapted alignment optimization algorithms are needed to deal with pure onset features.

In [Ewert and Müller, 2009], the approach of using note onset information for audio alignment was developed further. To obtain a feature which is not only robust but also allows for an accurate onset detection, the strengths of each pitch's onset candidates within a window of 20 ms are added up. After applying a logarithm, a Chroma representation of these onset indicators is obtained by taking the sum over all values representing an equal pitch class. To account for dynamic changes, the chroma onset values are normalized with respect to the maximum value within a sliding window. In a final step, the individual, sparse Chroma onset peaks are given a temporal spread. To this end, each peak is repeated 10 times after multiplying the original value with a respective decay factor.

The introduction of this decay does not reduce the temporal accuracy, but, on the other hand, improves the feature's robustness. While the sharp attack at the note onset time remains unchanged, the decaying repetitions will mask spurious, subsequent peaks of small values when considering a single pitch class. Also, a temporal spread of individual notes' representations results in frames where all notes of a chord are present, independent of small asynchronies.

A modification is the combination of this onset-based descriptor with standard Chroma vectors. In doing so, notes whose onset was missed will nevertheless be represented in the combined feature due to their contribution to the Chroma values. In [Ewert and Müller, 2009], a simple sum of the two alignment costs corresponding to the two features was reported to outperform the individual features. In addition, [Arzt et al., 2012] found that by combining the principles of onset-based descriptors and Chroma vectors adaptive to the signal novelty within a short sliding window, results can be improved even further.

3.4. Pitch Activation

While Chroma vectors and onset-based features are adequate descriptors for the tonality of an audio frame and for signal novelty respectively, they do not reveal details at the pitch level. This is not only tolerable when global alignments up to a certain required

accuracy are considered, but in fact a deliberate approach to increase the robustness of the features by avoiding the error-prone task of multi-pitch extraction. However, Table 2.6 and Table 2.7 show that asynchronies between individual notes of a chord are significant and cannot be ignored when alignment accuracy is an issue.

In [Niedermayer, 2008], we have proposed a spectrogram factorization based *Pitch Activation* feature in the context of automatic music transcription. Later (see [Niedermayer, 2009a]) we adapted this feature for the post-processing of individual notes' onset times. While the underlying principle is similar to non-negative matrix factorization, the specific strength of the Pitch Activation feature is that the musical context can be taken into account. This is particularly valuable for the local refinement of Audio-to-Score Alignments, where the notes which are played within a certain range of time are already known with a relatively high confidence.

3.4.1. Non-negative Matrix Factorization

As described in [Niedermayer, 2008], the principle behind the Pitch Activation feature is derived from the idea behind non-negative matrix factorization (NMF) which was introduced in [Lee and Seung, 1999] in the context of image processing. There, a number m of facial images is represented by an n dimensional column vector of non-negative pixel values each. The resulting matrix V of the size $m \times n$ is decomposed into two as well non-negative output matrices W and H of size $m \times r$ and $r \times n$ respectively, such that

$$V_{i,j} \approx (W\,H)_{ij} = \sum_{a=1}^{r} W_{ia} H_{aj} \tag{3.30}$$

where the columns of W are called *basis images* whereas each column of H corresponds to an image vector in V and is called an *encoding*. The factorization's rank r is chosen such that $(n+m)r < nm$, i.e., such that a reduction of data is achieved.

Since perfect factorization is not possible in almost all cases, a solution to Equation 3.30 with minimal error of reconstruction is achieved by minimizing a cost function over the difference between V and $W\,H$. Common such objective functions are the Euclidean

distance $E(V, WH)$ or the (generalized) Kullback-Leibler divergence $D(V \parallel WH)$ given as

$$D(V \parallel WH) = \sum_{ij} \left(V_{ij} \log \frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij} \right) \tag{3.31}$$

The concept of decomposing data matrices while minimizing the reconstruction error is not specific to NMF. However, other approaches impose different constraints on the factorization. Vector Quantization, for example, uses a unitary constraint on each encoding vector $H_{\cdot j}$, i.e., each image is encoded as its best matching prototype. Another example is Principal Component Analysis where the atoms in $W_{i\cdot}$ are forced to be orthonormal and the weightings in $H_{\cdot j}$ must be orthogonal. In contrast to Vector Quantization, this allows for an actual decomposition of signals into individual components. However, allowing negative pixel values in the basis images does not yield atoms which are intuitive or even represent meaningful partials. In the domain of audio signal processing, this would result in spectral patterns where individual frequencies could have negative amplitudes.

When we apply NMF to a power spectrum, as obtained by the short time Fourier transform, the basis components in W are weighted frequency groups that are found to sound together. Ideally, they belong either to a single pitch played on a certain instrument or a group of pitches that are normally played together, such as the notes of a chord. If the number r of basis components is smaller than the number of different pitches played, some of them have to be either omitted or grouped within one atom. In the reverse case, where r is sufficiently large, there can be atoms representing noise or there is more than one atom per one single pitch. It is very likely that one component represents the sustained part of a note whereas another maps to the note onset with much richer harmonics. A detailed investigation on these effects can be found in [Plumbley et al., 2006].

The component activation in H_{rj} expresses the strength of the r^{th} atom at time frame j. Due to the non-negativity constraint, the combination is additive only, giving consideration to the fact that there is nothing like a negative velocity of notes. Although power or magnitude spectra are not additive, assuming that they are a linear combination of their weighted components, i.e., the spectra of individual notes sounding concurrently, is an approximation which has been shown to yield acceptable results.

Effective algorithms for the calculation of the NMF have been introduced in [Lee and Seung, 2001]. Multiplicative update rules are used in order to find local min-

ima starting from randomly initialized W and H. Using the divergence from Equation 3.31 as cost function, these update rules are

$$H_{aj} \leftarrow H_{aj} \frac{\sum_i W_{ia} V_{ij} / (WH)_{ij}}{\sum_k W_{ka}} \tag{3.32}$$

$$W_{ia} \leftarrow W_{ia} \frac{\sum_j H_{aj} V_{ij} / (WH)_{ij}}{\sum_\nu H_{a\nu}} \tag{3.33}$$

Also, in [Lee and Seung, 2001] proof is given that the divergence is (i) non-increasing under the above update rules and (ii) invariant if and only if W as well as H are at stationary points, i.e., the algorithm converges towards a (local) optimum.

Several works like [Cont, 2006], [Plumbley et al., 2006], [Smaragdis and Brown, 2003], [Weiss and Bello Correa, 2010], [Vincent et al., 2007], or [Virtanen et al., 2008] have concentrated on applying matrix factorization using non-negativity and sparseness constraints on MIR tasks. Although NMF is a capable means to decompose a spectrogram into self-contained components, it has a number of drawbacks in the context of audio alignment and music signal processing in general.

One inherent problem of NMF based approaches is the determination of an appropriate number r of base components. While in the context of Audio-to-Score Alignment, the number of different pitches played during a specific piece of music can easily be obtained from the score, there is no guarantee that each played pitch is represented in the resulting dictionary W of atoms. [Plumbley et al., 2006] report that even when they used a more than sufficiently large number of base vectors in an NMF as well as in two sparse coding approaches, a small number of notes was not represented in the result. This is the case when chords or certain residual noise patterns become more significant than single tones that are played only very rarely.

Choosing a value r larger than the number of different pitches played increases the chance that each pitch has at least one corresponding atom, but, inherently results in higher computational costs. Also, the problem of mapping basis vectors in W to their corresponding pitches is likely to become harder proportionally to the excess of atoms.

In addition, learning a dictionary of independent components while aligning musical audio to a given score does not seem to be a natural way of approaching the problem. As shown in [Hainsworth, 2003], humans start by detecting the genre and style of a piece, which allows them to limit the number of possible instruments and timbres to be expected. Learning the dictionary of independent components along with their

activation is, as pointed out above, likely to model noise as well and therefore prone to over-fitting. Restricting dictionary vectors to feasible values, i.e., models of the tones which are expected to be played during the performance of a piece in advance is a reasonable means of preventing over-fitting as well as nonessential computational costs.

3.4.2. Non-negative Least Squares Factorization

To overcome the above described drawbacks of the NMF approach, we proposed to fix the dictionary W of atoms in [Niedermayer, 2008]. First, the number r of independent components is set to the number of possible notes regarding the pitch range of the certain instrument or the piece in focus. In addition, the single atoms are chosen to be stereotypical tone models of the corresponding pitches. A straightforward approach would be to use multiplicative updates on a random initialization of H applying the rule in Equation 3.32 while omitting Equation 3.33 and leaving W unchanged. However, a more efficient approach is to exploit the fact that W is fixed and Equation 3.30 can be resolved to

$$v_i \approx (\overline{W}\, h)_i = \sum_{a=1}^{r} W_{ia} h_a \tag{3.34}$$

where $\overline{W}$ is the fixed dictionary. v and h are column vectors representing one time frame of the spectrogram and the Pitch Activation respectively. In order to measure the quality of an approximation in Equation 3.34, again a cost function is needed. A convenient measure is the mean square criterion where

$$f = \frac{1}{2} \parallel \overline{W}h - v \parallel_2^2 \tag{3.35}$$

has to be minimized, while regarding the constraint of non-negativity. According to [Lawson and Hanson, 1974] this problem is solved by an iterative algorithm as follows.

1. Initialize all elements of h to zero and introduce two sets P and Z where P is empty and Z contains all indices within h.

2. Compute the gradient $\bigtriangledown_h f = \overline{W}^T \cdot (v - \overline{W}h)$ where f is the cost function as defined in (3.35).

3. If $Z = \{\}$ or $\forall i : i \in Z \Rightarrow (\bigtriangledown_h f)_i \leq 0$ then terminate.

4. Find the maximum element of $\bigtriangledown_h f$ and move its index from Z to P.

5. Solve the unconstrained linear least squares problem $\overline{W}_{sub} \cdot z = v$ where $\overline{W}_{sub}$ is a copy of $\overline{W}$ where all columns corresponding to indices in Z are set to zeros. Within the result z only those elements with indices contained in P are significant. The others are set to zero.

6. If $\forall i : i \in P \Rightarrow z_i \geq 0$ then z is a feasible solution, h is set to z and the main loop is continued at step 2.

7. If the above condition does not hold z can only contribute to the new temporary solution up to a certain amount. Therefore the factor α is calculated as $\alpha = argmin_i(h_i/(h_i - z_i))$ where only the indices of negative elements in z are allowed as i.

8. Calculate the new temporary solution using α from the above step as $h = h + \alpha(z - h)$

9. Move from P to Z all indices for which the corresponding element in h is zero. Continue the inner loop at step 5.

Although the result of one frame is a useful hint for the computation of the next frame, single time frames can now be processed independently. This makes the method not only suitable for parallelization but also for online processing. Reassembling the results of individual frames gives a complete activation matrix, such as H from Equation 3.30 as an optimal non-negative quotient of an input power spectrogram V and a given tone model dictionary W. The method can, therefore, be seen as a non-negative matrix division in contrast to the uninformed matrix factorization.

From another point of view, the vectors h are a feature, representing the strengths of the individual pitches within a chord. Within this work, it is called *Pitch Activation* and used to yield a quasi-transcription of an audio recording to be able to perform Audio-to-Score Alignment in the symbolic domain (as described in Section 4.3) and for the refinement of note onsets (as described in Section 6.2).

3.4.3. Tone models

The above described spectrogram factorization requires a set $\overline{W}$ of pre-trained tone models. Such a model consists of a prototypical spectrum of the same frequency resolution and scaling as the transformed audio frame v under consideration. A common approach (cf. [Bertin et al., 2010], [Vincent et al., 2008], and [Virtanen, 2007]) is to generate generic tone models based on the ideal frequencies of the partials of a note while allowing for a certain deviation. To this end, a cosine function in the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$, i.e., the positive part of the cosine wave, was chosen to represent a partial. The model of the tone with a fundamental frequency f_0 is then obtained as the sum over all partial models m_i defined as

$$m_i(f) := \begin{cases} h^i \cos\left(\frac{\pi}{2} \frac{f-(i+1)\,f_0}{\delta_s(f)}\right) & \text{if } f-(i+1)\,f_0 <= \delta_s((i+1)\,f_0) \\ 0 & \text{otherwise} \end{cases} \tag{3.36}$$

where $h \in (0, 1]$ is a decay factor and $\delta_s(f)$ is the frequency dependent tolerance range of s semitones calculated as

$$\delta_s(f) = f\left(1 - 2^{\frac{s}{12}}\right) \tag{3.37}$$

A frequency dependent tolerance range has the beneficial side-effect that this range becomes wider for harmonics of higher order, which implicitly accounts for increasing inharmonicities. Alternatively, [Vincent et al., 2008] describes a system where inharmonicities are explicitly taken into consideration. Then, instead of increasing the tolerance range, models are obtained as a series of constantly sharp peaks at the predicted detuned harmonic frequencies.

For vibrating strings, the index i runs from 0 to h_{max}. For wind instruments, in contrast, the model has to be adapted, such that only harmonics at uneven multiples of the fundamental frequency are included ($m = \sum_{i \in \{0,2,4,\ldots 2h_{max}\}} m_i$)[4]. An example tone model of an A4 for string instruments is shown in Figure 3.1.

Preliminary experiments in the context of audio alignment have shown that a decay factor of 0.8 and a maximum tolerance of one semitone yield good results. During the decomposition of a spectrum, the relatively wide tolerance range allows for frequency

[4]Remark: m_0 models the fundamental frequency (f_0), m_1 corresponds to the first harmonic ($2f_0$), and so on. Thus, the models of uneven multiples of the fundamental frequencies have even indices ($m_i \hat{=} f_0(i+1)$).

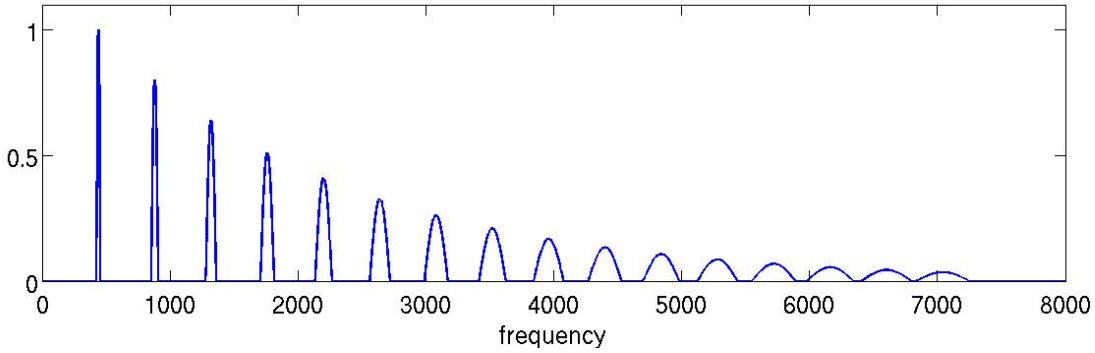


Figure 3.1.: Tone model of the note A4 (pitch 69, 440 Hz)

bins in between two adjacent pitches to potentially contribute to the activation of both tones. This seems to introduce unnecessary ambiguity. However, such ambiguities are not an issue due to a sufficient large number h_{max} of partials taken into account for each pitch.

While the generic model can easily be generated for arbitrary pitches, it will most certainly not contain typical characteristics of notes played on musical instruments. The fact that real tones do not satisfy the assumption of ideal harmonic frequencies has been know for many decades (see [Schuck and Young, 1943]). It was found that harmonic partials generally lie above their ideal frequencies by an amount quadratically increasing with their order. In [Fletcher, 1964] this effect is approximated by

$$f_n = f_0(n+1)\left(\frac{1+B\,(n+1)^2}{1+B}\right)^{\frac{1}{2}} \tag{3.38}$$

The parameter B is not only dependent on the exact instrument but also on the pitch of a note. While inharmonic deviations are least significant in the range between C2 and C4, they increase towards the lower and even more rapidly towards the higher pitches. MIR systems which explicitly take these inharmonicities into account are presented in [Klapuri, 2003] and [Vincent et al., 2008], for example.

A direct consequence of the inharmonicity is a "stretched" tuning of instruments. A common tuning process (cf. [Schuck and Young, 1943]) starts by tuning one octave according to a reference frequency and the ideal frequency relations between the individual tones. Then, the tones of the adjacent octaves are tuned such that no dissonance can be observed between the tone under consideration and its corresponding one within the reference octave. In doing so, the fundamental frequency of the higher tone will be the same as the first harmonic frequency of the lower tone, resulting in a propagation of

inharmonic frequency deviations to the fundamental frequencies of subsequently tuned tones.

[Niedermayer, 2008] proposes to use tone models which do not only consider an arbitrary number of harmonics, their assumed ideal frequencies, and an approximation of their relative strengths but also reflect the actual characteristics of the tone played on a specific instrument by a prior training phase. The necessary training data can consist of recordings of the single pitches played on the particular instrument or a similar instrument of the same class. For the actual training, the same principles as for the computation of the Pitch Activation feature are applied. Starting from Equation 3.30 again, instead of fixing W to a given dictionary, H is chosen to have $r = 1$ to do justice to the fact that there is only one note played. The single values of H are then set according to the amplitude envelope expressing the current loudness of the sound. The only remaining variable there is W. Due to $r = 1$, the dictionary W consists of a single atom w, for which an approximation can be obtained by applying the non-negative least squares criterion and the according optimization algorithm as described above.

In this way, an average model representing the entire sustain phase of a tone is obtained. Since higher order harmonics tend to decay relatively fast, using training data where tones are sustained for a long time results in these harmonics being averaged out. The note length of the training tones should, therefore, relate to the expected note durations found in music performances.

In cases where only recordings of some notes and not the whole pitch range are available, interpolation is applied. Given a fundamental frequency f_0, for which the tone model is not known, the starting points are the two nearest fundamentals f_l and f_h with known energy distributions such that $f_l < f_0 < f_h$. The interpolation is then done in two steps. First, center frequencies f of the unknown spectrum's bins are mapped onto the known spectra by linear scaling, i.e., such that the fundamental of f_0 is mapped to the fundamentals of f_l and f_h, the first harmonic is mapped to the respective first harmonic frequencies, and so on. The mapped frequencies f' are, therefore, obtained as

$$f' = f\frac{f_i}{f_0} \qquad \text{with } i \in \{l, h\} \tag{3.39}$$

In a second step, the energy has to be obtained for each bin. Due to the mapping computed in the first step, the frequency f' corresponding to a bin's center frequency f is known. However, since all spectra are discrete in the frequency dimension, f' is likely to lie at some position between two center frequencies. Therefore, the energy $y(f')$ of f' is approximated applying Stirling's formula for interpolation. To this end,

the energies y_0, y_{-1}, and y_{+1} at the bin with the center frequency $f_{nearest}$ closest to f' and its two adjacent bins are considered. $y(f')$ is then given as

$$y(f') = y_0 + \frac{y_{+1} + y_{-1}}{2} \cdot d + \frac{y_{+1} - 2y_0 + y_{-1}}{4} \cdot d^2 \tag{3.40}$$

with

$$d = \frac{f' - f_{nearest}}{\delta f} \tag{3.41}$$

where δf represents the constant difference between the center frequencies of two adjacent bins. In a final step the approximations using the lower and the upper frequency model are combined by taking the average.

Two sets of training samples of piano tones were obtained from the Bösendorfer SE 290 computer controlled grand piano and from the online database of tone samples of various musical instruments of the University of Iowa [5]. The Bösendorfer samples were recorded in 2001 by Werner Goebl (cf. [Goebl, 2003]) for every fourth pitch, starting at MIDI pitch 12, at five different velocities (30, 50, 70, 90, and 110). For pitches up to C2, i.e., MIDI pitch 36, recordings of a length of 9 seconds were available. For the higher pitches, a recording length of 4 seconds was chosen due to the faster decay of the tones.

The musical instruments samples of the University of Iowa were recorded in 2001 on a *Steinway & Sons* grand piano. Samples are available from B♭0 to C8, i.e., MIDI pitches 22 to 108, in three different loudnesses (*pp*, *mf*, and *ff*). Each tone was sustained for about 2 seconds and is preceded and followed by silence.

Samples from both sources were available in stereo at a sampling rate of 44.1 kHz and a 16 bit quantization. Respective tone models were obtained as described above. A comparison between these trained models (see Figure 3.2 for an example) and the ones obtained by mathematically modeling the harmonics at their ideal frequencies (see Figure 3.1) was done in the context of anchor note extraction as described in Section 6.2. Although models trained from tones of the same piano as was used for the performance resulted in a marginally higher alignment accuracy, we decided to use the generic models in our system for generalization reasons. The Pitch Activation for the beginning of the first movement of the sonata k.279 is shown in Figure 3.3 where only the models of those notes which are expected to be played within the shown search window are used for the factorization.

[5] http://theremin.music.uiowa.edu/MIS.html

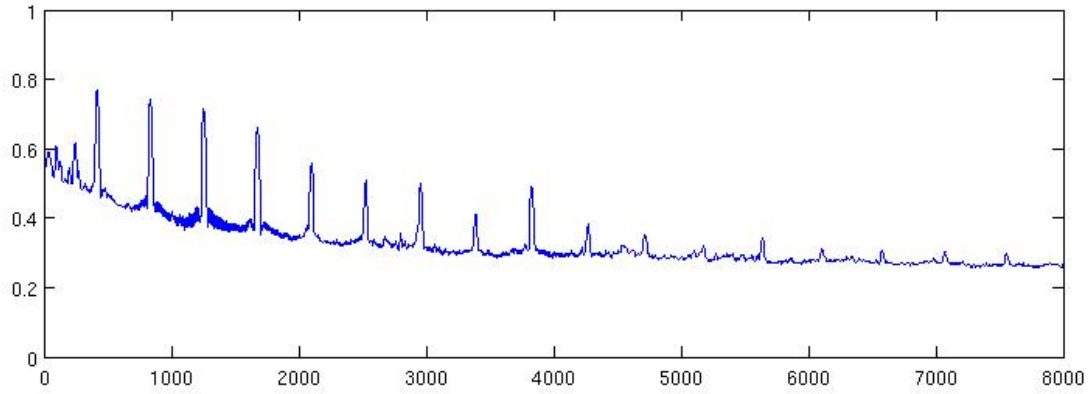
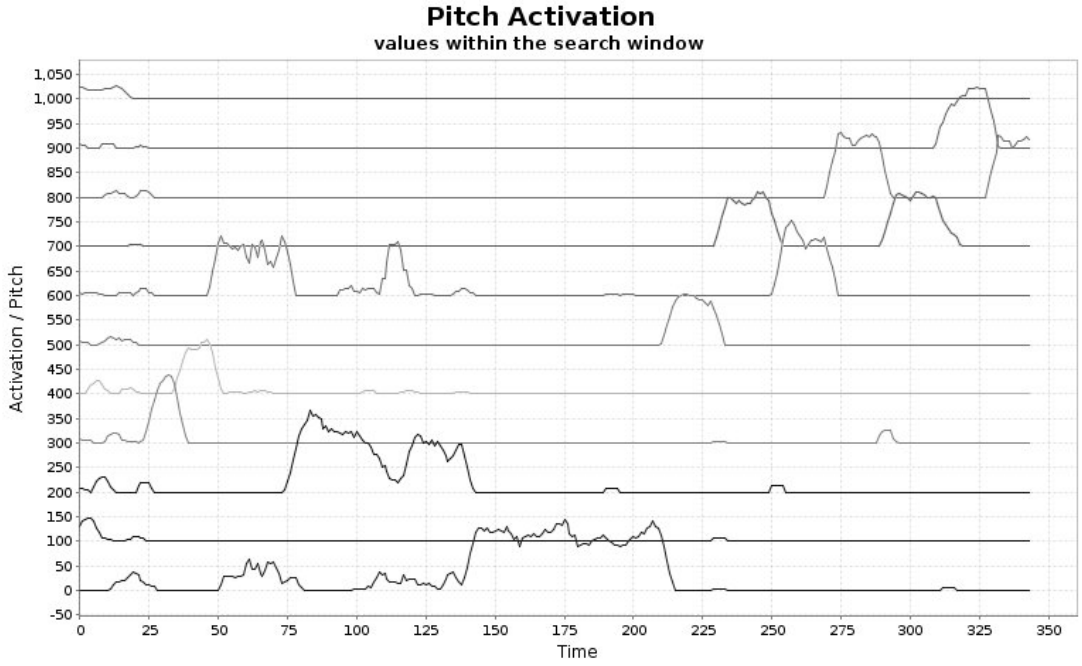
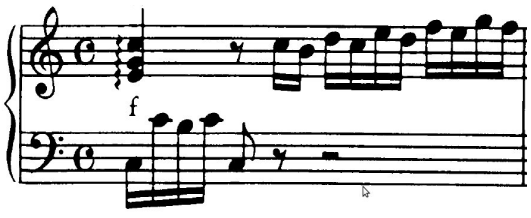


Figure 3.2.: Tone model of the note A♭4 trained from a sample recording (pitch 68, 415.3 Hz)



(a)



(b)

Figure 3.3.: Pitch Activations of the beginning of the first movement of the sonata k.279, where the activations of the individual pitches are plotted with an offset of 100 on the value axis (a) and the corresponding score (b)

3.5. Extraction of Score Features

Audio-to-score alignment approaches are divided into two main classes. One, where the score is represented as a graphical model, such as a *Hidden Markov Model* (HMM), and the other one, where the score is represented by the same features or symbolic descriptors as extracted from the audio recording. While the modeling of a score by means of HMMs or similar representations is described in Section 4.2 together with the decoding of such models, the extraction of the three audio features described above from score MIDI files will be discussed here.

A straightforward approach for obtaining audio features from a symbolic representation is to synthesize the score and to follow the same extraction process as for the actual audio recordings. Due to pre-processing steps, such as spectral whitening and the generally significant data reduction performed during feature calculation, the requirements concerning the quality of instrument samples and naturalness of the rendering are moderate. In our experiments, we used the freely available software synthesizer *timidity++* (see Section 2.4.1) and its, as well free, default sound bank.

The seeming drawback that high level, symbolic information is lost, does also have a number of benefits. Most important, systematic sources of error during the computation of audio features, such as inadequate or simplified handling of harmonics, overlapping partials, inharmonicities, etc., are reproduced during the score feature extraction. One example is the Chroma feature, where the second harmonic contributes to the pitch class a perfect fourth, i.e., five semitones, below the one corresponding to the fundamental frequency of a note (see Table 3.1).

In addition, calculating score features on audio renderings results in timbre information being integrated into the feature values. While the differentiation between various instruments is not an issue, realistic decays of individual partials yield values more similar to those obtained from the actual audio data.

On the other hand, rendering each score prior to feature extraction is expensive in terms of computational costs. Since the improvements due to calculating score features in the same manner as audio features are not substantial, these costs cannot be justified in every context. Therefore, [Dannenberg and Hu, 2003] first proposed to calculate Chroma vectors directly from the MIDI representation of a score. In a preliminary experiment, the authors showed that the Chroma feature is relatively insensitive to timbre. The score of a classical orchestral piece was synthesized using (i) the original instruments and (ii) a generic piano sound only. The resulting alignments to actual recording of the performances showed only marginal deviations. Based on this conclusion, it is also possible to obtain a Chroma vector by simply representing each present pitch by

the value 1, taking the sum for each pitch class and performing a normalization. In [Hu et al., 2003], it is shown that, even then, there is little impact on the alignment results. In this work, a slightly modified algorithm was applied, which also considers note onsets and offsets during a single frame and down-weights the contributions of the respective pitches accordingly.

The Pitch Activation feature can be directly extracted from the score in a similar manner. While a simple approach would be to set the activation of each pitch present at a certain frame to 1 and the ones of the silent notes to 0, introducing a decay factor better reflects the actual sustain phase of a tone. For the accurate Audio-to-Score Alignment system proposed in this thesis, the Pitch Activation feature is used to obtain precise onset estimates in the refinement step. Therefore, Pitch Activation values computed from scores are only needed in our symbolic domain Audio-to-Score Alignment system (see Section 4.3) which was designed to reduce computational costs of the alignment step. There, in accordance with this objective, the binary feature representation was chosen.

3.6. Conclusions and Consequences for this Thesis

In the literature, Chroma features in conjunction with DTW are a common choice for audio alignment tasks due to their high robustness against a number of aspects such as timbre or room acoustics. Nevertheless, such a system can generally not compete with uninformed state-of-the-art onset detectors in terms of accuracy. On our evaluation corpus, about 74% of all note onsets obtained by DTW and Chroma vectors were within a 50 ms tolerance range around the actual timings. About 32% of the onsets were accurate applying a 10 ms tolerance range.

Based on the motivation of improving alignment accuracy, while not trading it against robustness, features similar to Chroma vectors including explicit onset indicators have been developed. Preliminary experiments using our own implementations of several of the above described onset based features – also in a mixture with Chroma vectors – revealed the ones described by [Arzt et al., 2012] (again, in conjunction with DTW) to perform best on our evaluation corpus. However, it became apparent that these features, despite yielding accurate results on most pieces, fail at some passages where the performer has inserted many additional notes which are not in the score. To be specific, this was the case for the pieces k.457-2 and k.284-3 where the error was the largest (and also for the piece k.475-2 due to a similar reason). There, about 900 notes were aligned with a temporal deviation from the actual onsets between 10 and 18 seconds before the alignment became accurate again.

The objective of this thesis is to develop an alignment system, where the user has to correct as few notes as possible to obtain an accurate annotation. Therefore, we argue that as soon as the timing of an aligned note is obviously not correct, the actual temporal deviation up to a certain extent is of less interest. Timing errors of more than 10 seconds where a manual correction exceeds a simple refinement, however, can be considered as too severe. In the final Audio-to-Score Alignment system proposed in this thesis, we therefore apply Chroma vectors instead of a mixture including onset based features due to their robustness. Here, the largest alignment error in the above mentioned piece k.284-3 does not exceed 2.5 seconds (see Appendix B, for details).

Nevertheless we want to give an explanation of the large errors produced by the, otherwise very accurate, onset based methods. At each of the three pieces where the alignment fails locally, the performer plays a large number of additional notes which are not in the score. To be specific, in the piece k.284-3 51 notes are added within 4 consecutive measures as part of additional trills. For the performances of the other two pieces the situation is similar (k.457-2: 60 additional notes within 2 measures due to two glissandi; k.475-2: 39 notes of a glissando within one single measure, where the glissando is notated as a chord). Even when also taking harmonic content into account, the attempt to match such large numbers of additional onsets to notes within a score will inherently result in significant errors.

We also experimented with Pitch Activation as a feature for a basic alignment. Although the quasi-transcription which can be derived from the feature values is an adequate means to reduce the amount of data processed during the alignment step, it became obvious that the note detection is not robust enough to yield accurate alignment results (see Section 4.3). Nevertheless, by tuning the feature extraction to the local musical context of a note based on an initial alignment we were able to extract refined onset times. We initially proposed this method in [Niedermayer, 2009a], showing that it outperforms the initial alignment as well as a reference onset refinement technique based on selective bandpass filtering. In Chapter 6, we describe how a slightly adapted version of the Pitch Activation feature is used in the system we propose here.

4. Audio-to-Score Alignment Techniques

Based upon the feature extraction step as described in Chapter 3, the actual Audio-to-Score Alignment can be performed. Such an alignment is a symmetric binary relation between the set T_a comprising the timestamps of all audio frames and the set T_s of timestamps of all score frames or score events. However, due to the discrete nature of digitized data, a compression or stretching of the two time scales relative to each other will inherently inhibit the relation from being either bijective or a function.

An important consequence of mapping timestamps with respect to such a relation is that notes which are notated concurrently will inherently be assigned a uniform onset time within the audio recording. This is a major drawback which is not resolved in the vast majority of current state-of-the-art systems. It is the main contribution of this thesis, to present a system which allows for the distinction of individual notes within a chord. This issue will be topic of Chapter 6.

In the following, three methods for the timestamp mapping will be described. The first one is based on acoustic features, calculated from the score in the same manner as from the audio signal. As described for the Chroma vectors, this can by done by either calculating an idealized feature directly from the MIDI data or by synthesizing the score. The alignment is then obtained by comparing individual features of the two sequences to each other and identifying corresponding frames by minimizing an accumulated matching cost measure.

A second method is to construct a graphical model from the score. In doing so, information contained in the symbolic data is not reduced during the feature calculation, but can be incorporated into the model in almost arbitrary diversity. Distinguishing between attack, sustain, and decay phase of a note is in principle possible as well as defining a specific probability distribution over possible lengths of a note. Aligning the audio signal to the score is then carried out by decoding the model, i.e., computing the state sequence which is most likely to produce the observed output.

The third approach is also based on the idea of exploiting the symbolic representation of the score. But instead of relating score events to acoustic descriptors extracted from the audio signal, the score is directly aligned to a quasi-transcription of the audio recording. This rudimentary transcription is obtained by reporting a note during the time where the Pitch Activation feature of the respective pitch is greater than zero.

While the first two approaches are common in the literature, the third one – working in the symbolic domain – has been neglected so far. To the author's knowledge, the only related system was described in [Müller et al., 2004] and evaluated in terms of computational costs. In [Niedermayer, 2009b], we presented the first quantitative evaluation of the alignment accuracy of such a system.

4.1. Dynamic Time Warping

Dynamic Time Warping (DTW) is a general technique for finding a globally optimal alignment between two time series. It assumes that both series contain the same or very similar semantic content, which is, however, non-linearly stretched or compressed, i.e., warped, along the time axis. DTW is, therefore, predestined to be applied in the context of audio processing to establish a connection between two instances of the same words or melodies which are spoken, sung, or played at different, varying tempos.

Although many variants and modifications of Dynamic Time Warping exist, they rely on a common basic procedure. At first a dissimilarity matrix of the two sequences is calculated, which contains the isolated costs of alignment for each possible pair of elements. Based on this information, the optimal alignment is obtained as the sequence of pairs which yield the minimal sum of costs.

4.1.1. Similarity Measure

In the audio alignment literature, the Chroma vector is the most commonly used feature (see, for example, [Dannenberg and Hu, 2003], [Hu et al., 2003], [Joder et al., 2010b], [Müller et al., 2005], [Serrà et al., 2008], [Niedermayer and Widmer, 2010a], and [Niedermayer, 2009a]). According to this, the time series describing a specific audio recording is a series $A = \{a_i\}$ with $i \in [0, N-1]$ and $a_i \in \mathbb{R}^{12}$. Given the hop size l_{hop} of the time-frequency transform used to obtain the Chroma feature, the index i corresponds to the time $t_i = i \frac{l_{hop}}{f_s}$.

When computing the sequence B of score features, it is beneficial to stretch the score to the performance time of the audio recording beforehand. In doing so, individual frames are likely to contain similar amounts of information.

Based upon the two sequences A and B, the dissimilarity matrix D is calculated by comparing each frame of the audio recording to each frame of the score representation as

$$D_{ij} = C(a_i, b_j) \quad \text{, for all } a_i \in A \text{ and } b_j \in B \tag{4.1}$$

The alignment cost function C is chosen with respect to the type of feature used for the comparison. Common choices for C in combination with Chroma vectors are the Euclidean distance on the normalized feature values and the cosine distance (cf. [Niedermayer and Widmer, 2010a], [Müller et al., 2006], [Joder et al., 2010a])

$$C_{cos}(a_i, b_j) = \frac{a_i b_j}{|a_i|\,|b_j|} \tag{4.2}$$

4.1.2. Minimal Cost Calculation

Aligning two feature sequences is equivalent to finding a path through the dissimilarity matrix D. Each single mapping of a frame a_i within the audio feature sequence to an arbitrary score frame represented by b_j can be interpreted as a point (i, j) and has its corresponding element D_{ij} describing the cost at this point. Connecting the matrix elements while mapping subsequent frames yields an alignment path. However, in order to obtain meaningful results such an alignment path has to meet several constraints.

Continuity The constraint of continuity forces a path to proceed through adjacent cells within the dissimilarity matrix. Jumps would be equal to skipping frames without considering the costs of this operation.

Monotonicity The constraint of monotonicity in both dimensions guarantees that the alignment has the same temporal order of events as the reference sequence.

End-point constraint The end-point constraint forces the beginning and the end of the path to be the diagonal corners of the dissimilarity matrix. In doing so it is assured that the alignment covers the whole sequences.

Even within these constraints, an exhaustive search over all possible alignment paths would exceed manageable computational efforts. Therefore, a dynamic programming approach is applied, where partial alignment paths of increasing length up to a certain element D_{ij} are considered iteratively. The algorithms starts at the point $(0, 0)$ and rates this degenerated alignment path, consisting of one way-point only, with the cost $D_{0,0} = C(a_0, b_0)$. Then, still within the initiation phase, all partial alignments ending at a point $(i, 0)$ or $(0, j)$ are considered. The calculation of their respective costs is straightforward, since the corresponding alignment paths are unique due to the monotonicity constraint. The minimum costs of paths ending at other points (i, j) can then be calculated in a recursive manner according to

$$Accu(i,j) = \min \begin{cases} Accu(i-1, j-1) & + C_{ij} * w_d \\ Accu(i-1, j) & + C_{ij} * w_s \\ Accu(i, j-1) & + C_{ij} * w_s \end{cases} \tag{4.3}$$

The three options correspond to partial paths ending with a diagonal step, an upwards step, and a step to the right within the dissimilarity matrix D. In addition to the actual local distances, weights w_d and w_s are introduced to adjust the algorithm's preference towards diagonal steps. Setting both weights to 1 results in smoothed alignment paths since diagonal paths traverse only half as many points as ones consisting of steps to the right and upwards only and are, therefore, much more cost efficient. In [Niedermayer, 2009a], we have used the values 1.4 and 1.0 for w_d and w_s which still gives diagonal steps a preference over horizontal or vertical ones.

When one is only interested in the optimal alignment, the algorithm can be implemented such that the cost calculation works in-place, i.e., the values D_{ij} are overwritten by $Accu(i, j)$ in order to save memory space. Alternatively, when one computes the Accumulated result line by line, a memory efficient method is to only store the current line and its predecessor.

The algorithm terminates when the costs of the complete optimal path are, according to the end-point condition, obtained as $Accu(N-1, M-1)$, with $N = |A|$ and $M = |B|$ being the lengths of the feature sequences. Due to the constraints on the alignment path, the result is guaranteed to be a global minimum. However, since two ore more of the terms in Equation 4.3 can be equal, the optimum does not need to be unique.

4.1.3. Path Backtracking

Since one is generally more interested in the optimal alignment path than its overall cost, the optimal path is reconstructed in a separate backtracking step. One point which must inherently lie on this path is the end-point $(N-1, M-1)$. To be able to reproduce the respective preceding way point, a second matrix is built during the forward step, memorizing whether the last step leading to a point (i, j) was diagonal, upwards, or to the right, i.e., which of the three options of Equation 4.3 was chosen. By tracking this information back to $(0, 0)$ the complete optimal path is obtain in a computationally efficient way. A dissimilarity matrix computed from a rendering of the mechanical score of the first movement of Mozart's sonata k.279 and the recording of a respective performance is shown in Figure 4.1.3. The alignment cost measure was chosen to be the cosine distance between Chroma vectors. Deviations of the (smoothed) alignment path from a strict diagonal show expressive tempo changes.

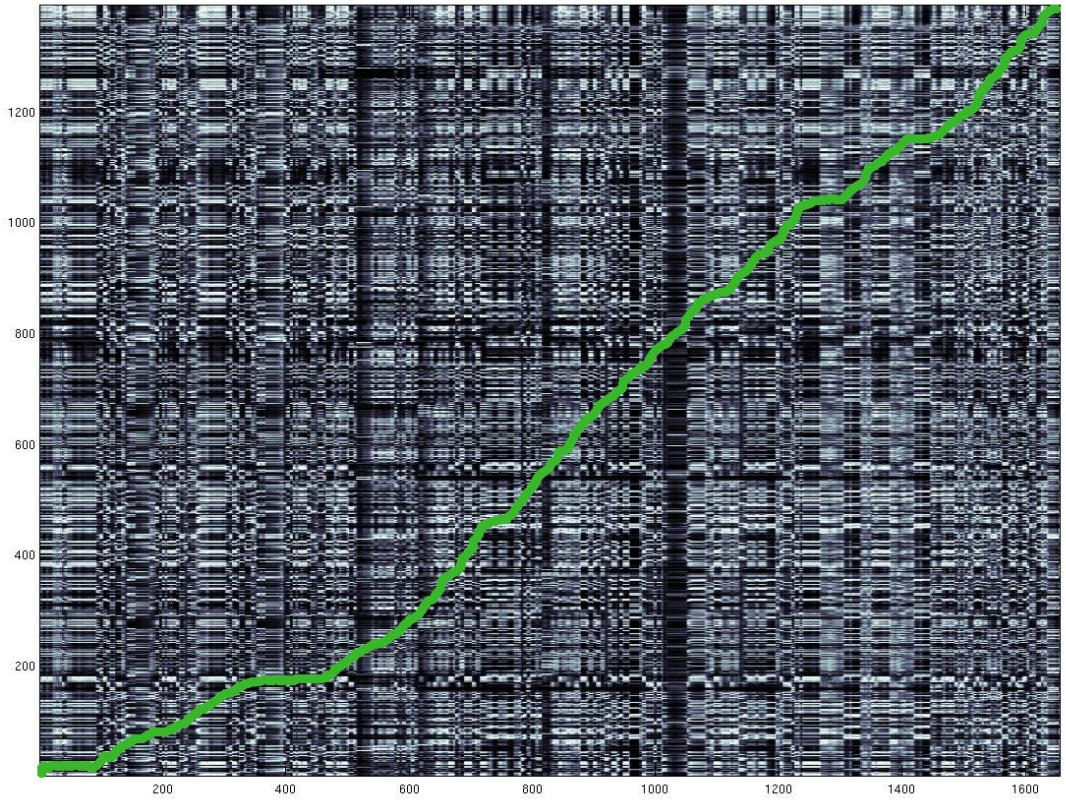


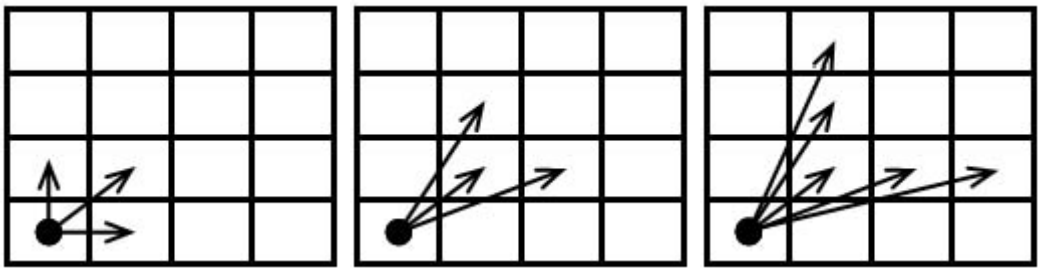
Figure 4.1.: Alignment cost matrix and path comparing the score (horizontal axis) of the sonata k.279 and a respective performance (vertical axis), where the units along the axes are audio and score frames and the respective matching cost is mapped onto the gray-scale

4.1.4. Enhancements of the DTW Algorithm

Especially in classical music, rapid tempo changes are a common means of artistic expression. Tempo fluctuation by a factor of 2 or 2^{-1} are not an exception. To account for this fact, the repertory of possible steps through the dissimilarity matrix D formalized in Equation 4.3 can be extended. By seemingly softening the continuity constraint and introducing a respective weight $w_{(n,m)}$, steps of length $|(n,m)|$ and slopes m/n are possible. The generalized rule then becomes

$$Accu(i,j) = \min_{(n,m)} \left\{ Accu(i-n, j-m) + C_{ij} * w_{(n,m)} \right\} \tag{4.4}$$

Although way-points of an alignment path are not necessarily adjacent cells, the continuity constraint is still met, since there is no gap between the start of one step and the end of its predecessor. Possible step constraints are shown in Figure 4.2, where (a) corresponds to the standard options as described above. In (b), steps towards the right and upwards are not allowed. This limits the resulting paths similar to the Itakura-parallelogram. While the maximal slopes of 2 and 1/2 account for tempo changes by a factor of 2, (c) depicts a set of possible steps where the tempo deviation accounted for has a factor of 3. Such extensions of the step constraints are used in [Macrae and Dixon, 2010], [Meron and Hirose, 2001], and [Soulez et al., 2003], for example.



(a) (b) (c)

Figure 4.2.: Possible constraints for steps allowed during a DTW computation

Another issue are silences at the beginning and the end of an audio signal and the respective score representation. While it is very likely that within an audio recording the first note onset will not occur with the first analysis frame, MIDI notations of a score often start without any offset at a tick number of 0. Since each alignment path has to start at the point $(0,0)$, this would inherently result in the first note of a piece being aligned to the first audio frame. To prevent this scenario, an arbitrary number of

frames of silence is inserted at the beginnings and endings of the two feature sequences. Justified by observations regarding common lengths of such pauses in audio recordings, we use a value approximating a duration of 2 seconds in our system.

4.2. Graphical Score Models

The Dynamic Time Warping approach described above works on two time series of audio features. This has the advantage that the method can be applied for audio-to-audio matching without any modifications. On the other hand, due to the extraction of audio features from the score, high-level symbolic information is lost or at least distorted. Also, DTW can be interpreted as a graphical model, where cells of the dissimilarity matrix correspond to states and the cost measure is the equivalent to the output probability of a state. However, in comparison to the DTW approach graphical models allow for more general state transitions and specially modeled output probabilities.

[Cano et al., 1999] propose *Hidden Markov Models* (HMMs) for score to performance matching. In [Raphael, 1999], a system which is also based on HMMs, for the offline as well as online segmentation of an audio recording of a monophonic piece into note objects given by a score, is presented. Hidden Markov Models are well known from the field of speech recognition (see [Rabiner, 1989], for example). They are based on the assumption that a series of observations is generated by a hidden state Markov process, i.e., a process where the next state depends solely on the current state and an a priori known state transition model. The application to Audio-to-Score Alignment is obvious, since the notes which are currently played, i.e., the states, are not directly observable from an audio recording of a musical performance. Instead, one can obtain a probability distribution for each state over all possibly observed outputs. Those outputs are modeled to be values of audio features which can be obtained from the signal.

4.2.1. Note and Chord Duration Modeling

Before a note's actual duration is modeled, some thoughts shall be spent on how a score note is represented in a Hidden Markov Model at all. The simplest method is to convert each note or each chord respectively into one state of an HMM. If the structure of the model is strictly linear and therefore requires each state to be visited this implies that each score event is assigned a unique time stamp within the audio recording. Otherwise, if the model structure allows states representing individual notes or chords to be skipped by introducing additional transitions, note durations are (as in the DTW

approach) not constrained at all. Skipped states result in notes being "merged" to chords whereas visiting a state and possibly following self-transitions corresponds to sustaining a note.

Following a hierarchical approach, notes can be modeled by an attack, a sustain, and a release phase, as proposed by [Cano et al., 1999] and [Orio and Déchelle, 2001], for example. This is justified by the fundamental differences in the expected observations during these phases. While, on instruments such as the piano, the note attack is characterized by a transient increase of energy throughout the whole spectrum, spectral patterns remain relatively steady while a note is sustained before partials are rapidly decaying during the release.

What is left to model is the actual duration of the sustain phase. Due to the assumption that the generating process is a Markov process, it is not possible to either introduce a likely note length, i.e., a number of iterations where the HMM stays in the same state, nor to enforce a minimum or maximum duration of a note, by means of a single state. [Raphael, 1999] provides two possible solutions to this problem. One approach is to model a note by a number of states equal to the maximum number of audio frames the note can be sustained. The states are connected linearly with no self transitions but allow for an additional short-cut from each node to the release state. The probability that the next state k is visited is then given as

$$p_k = P(T > k | T > k - 1) \tag{4.5}$$

where T is a random variable representing the duration of the respective note in units of audio frames.

Although it is intuitive and allows to model arbitrary probability distributions over the duration T of a note, the main drawback of the method described above is the large number of required states. A more efficient, however not as flexible, approach is to model the note duration by a number N of states equal to the allowed minimum of T. The states are, again, linearly connected, but, in contrast to the model described above, have self transitions with a constant probability of p instead of the short-cuts to the release state. The note duration T then follows a negative binomial distribution, given as

$$P(T = t) = \binom{t-1}{N-1} p^{t-N} (1-p)^N \tag{4.6}$$

for $t \geq N$.

[Cont, 2010] also describes the combination of these two ideas, where a number N of linearly connected nodes have self transitions which are chosen at a constant probability of p and short-cut transitions to the release state chosen at a constant probability of q. The resulting topology is shown in Figure 4.3 The remaining probability of the transition to the subsequent state is then $(1 - p - q)$ and the probability for $T = t$ follows the compound distribution

$$P(T = t) = \sum_{n=1}^{N} \binom{t-1}{n-1} p^{t-n}(1-p-q)^{n-1}q + \binom{t-1}{N-1} p^{t-N}(1-p-q)^{N-1}(1-p) \quad (4.7)$$

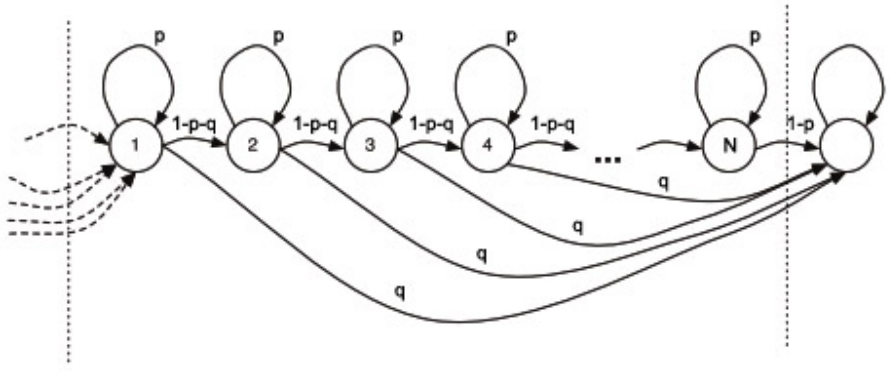


Figure 4.3.: Topology of an HMM for note duration modeling [Cont, 2010]

Instead of using this general HMM, [Cont, 2010] describes a Semi-Markov Model in the context of score following, where notes or chords are described by single macro-states following a Markov process while note durations are modeled by an explicit probability distribution.

4.2.2. Tempo Modeling

[Raphael, 2006] proposes to not only represent the relative durations of score notes, but to include the current tempo of the performance into the model. The timing of a note is then determined by the combination of time-varying tempo and local note-dependent temporal deviations. Such deviations can be due to the voice-lead phenomenon as

described in Section 2.4.2 or intended arpeggiations. The random variable representing the timing T_k of the k^{th} score event is then

$$T_k = T_{k-1} + l_k S_k + \tau_k \tag{4.8}$$

where τ_k is the local deviation, l_k is the score time difference between the k^{th} event and its preceding one, and S_k models the tempo process as

$$S_k = S_{k-1} + \sigma_k \tag{4.9}$$

The tempo change σ_k follows, as well as τ_k, a normal distribution with a mean of zero. Considering a score of a length of K distinct events, the probability of a certain tempo trajectory $s = (s_0, \ldots, s_{K-1})$ and an instance $t = (t_0, \ldots, t_{K-1})$ of all event timings is given by

$$p(s,t) = p(s_0)p(t_0) \prod_{k=1}^{K-1} p(s_k|s_{k-1})p(t_k|t_{k-1}, s_k) \tag{4.10}$$

Due to the assumption that the tempo changes σ and the local timing displacement τ both follow a normal distribution, $p(s,t)$ can be calculated based on estimates of the respective parameters and the probabilities $p(s_0)$ and $p(t_0)$ of the initial tempo and the first note onset respectively.

To obtain a joint model, also considering observations from the audio file, the tempo and timing processes are put into relation to the probability $p(y_n|x_n)$ as described above, where y_n is the audio feature calculated from the n^{th} frame and X_n are the (hidden) model states. The problem is that while $p(s,t)$ is calculated in the domain of score time, $p(y|x)$ is obtained from the audio recording. However, the state trajectory x and the sequence of note timings t can easily be transformed reciprocally. A joint probability can therefore be expressed as

$$p(s,x,y) = p(s,t,x,y) = p(s,t)p(y|x) \tag{4.11}$$

4.2.3. Observation Probability Distribution

The observation probability distribution $p(y_n|x_n)$ is the equivalent to the dissimilarity or cost measure used by the Dynamic Time Warping algorithm. In the Hidden Markov model literature these probability distributions are often called $b_i(v)$ (see [Rabiner, 1989]), where i is the index of a state x_n and v is the observation y_n. The dependency on the time denoted by the frame number n can be dropped due to the basic assumption that an observation solely depends on the current state.

The individual probabilities of a frequency v to be observed while the model is in state b_i can be modeled as a mixture of Gaussians $G(.; ., .)$

$$b_i(v) = \sum_{k=k_1}^{K} c_{ik}\, G(v; \mu_{ik}, S_{ik}) \tag{4.12}$$

where μ_{ik} is the mean vector, S_{ik} the covariance matrix, and c_{ik} the mixture coefficient with $\sum_{k=1}^{K} c_{ik} = 1$. These parameters can either be learned from training data by calculating the maximum likelihood estimators, or set manually, to incorporate musical knowledge.

4.2.4. Modeling of Asynchronies

As it is the case for Dynamic Time Warping, Hidden Markov Models, as described above, can not efficiently represent asynchronies between individual notes of a chord. A series of linearly connected states does not account for the uncertainty whether the notes are actually played simultaneously or which of the notes is played first. [Devaney and Ellis, 2009] describe a hierarchical HMM where not the notes or chords are detailed by micro-states, but the chord transitions.

Each transition between one note and its subsequent one is divided into a phase where the first note sounds, a transition or silence phase, and a span of time where the second note is present. Assuming a chord transition where each chord has a degree of polyphony of n, this results in n individual note transitions requiring 3^n states.

While the approach is feasible for pieces played by a small number of monophonic instruments or instruments with limited polyphony, it will become too expensive in terms of computational costs due to its exponential asymptotic complexity. At a degree of polyphony of 4 – which is relatively common for piano music (see Section 2.4.2) –

the number of HMM states modeling a singe chord transition is 81. For the highest degree of polyphony found within the evaluation data set used here, 8, to be specific, this number increases to 6561.

Additional complexity would arise from also accounting for sustained notes which outlast the onsets of several subsequent notes or chords. While this can easily be modeled for notes when their actual duration is known from the score, notes which sound for a long period of time due to the use of the sustain pedal of a piano are difficult to handle.

4.2.5. Model Training and Decoding

Once the topology of a graphical model is designed, parameters, i.e., probability distributions, defining the state transitions have to be determined. Doing so can either be achieved by exploiting prior knowledge about the underlying processes or by estimating these attributes from adequate training data. The final (Hidden Markov) model is then determined as $\lambda = (A, B, \pi)$, where A is the matrix of state transition probabilities, B defines the probabilities for specific observations to be made at a certain state, and the initial state distribution π (see [Rabiner, 1989]).

For the actual decoding of a Hidden Markov Model, i.e., the calculation of the most likely sequence of states, given the observations made, the *Viterbi algorithm* is used. This algorithms starts by calculating the most likely initial state, by

$$\delta_0(i) = \pi_i b_i(y_0) = \pi_i p(y_0|x_i) \tag{4.13}$$

Then, following a dynamic programming approach, the most likely paths up to the next observation are calculated as

$$\delta_t(j) = \max_{1 \leq i \leq K} \left[\delta_{t-1}(i) a_{ij}\right] b_j(y_t) \tag{4.14}$$

When the values for $\delta_{N-1}(j)$, corresponding to the states at the very last observation, are calculated, the most likely path, i.e., the *Viterbi path*, is obtained by backtracking the sequence of consecutive states. To this end, the index i of the state responsible for the maximum in Equation 4.14 is remembered in a separate matrix, as it is done in the Dynamic Time Warping algorithm.

4.3. Quasi-Transcription

While Dynamic Time Warping based on Chroma features, as most commonly described in the literature, relies on low-level descriptors for the audio as well as for the score representation, graphical models still use such low-level features to represent the audio recording, but, on the other hand, use a high-level description of the score. [Bloch and Dannenberg, 1985] originally described a third approach, where (in the context of an automatic accompaniment system) performance and score are matched in the symbolic domain. However, since the proposed system relies on symbolic input from keyboard-like devices its applicability is limited and the method cannot be directly transferred to Audio-to-Score Alignment.

In [Niedermayer, 2009b], we proposed a system based upon mid- to high-level descriptors of the audio content yielded by a quasi-transcription of the raw signal. To this end, the Pitch Activation feature is used as a mid-level representation corresponding to the pitches played. By setting note onsets at each time where a Pitch Activation rises above zero or another predefined threshold and defining the note offset as the point where the activation energy becomes zero again or has decayed by a certain factor, a high-level, symbolic description of the audio material is obtained.

This high-level audio description can be directly compared to the score, which is also available in a symbolic representation. Relating single events, i.e., note on- or offsets, instead of audio features sampled at a fixed rate reduces computational costs significantly.

4.3.1. The Symbolic Domain

Automatic Music Transcription for polyphonic pieces is still an open MIR problem. Common issues are confusions between pitches which share a number of partials, the detection of very low notes, and repeated notes, where the degree of spectral novelty is limited. In [Niedermayer, 2009b], we suggested accepting the fact that an automatic audio transcription process is error-prone and to account for possible errors by designing a specific cost measure when comparing the result to the score. While expecting a manageable amount of ambiguities between spurious note detections and correct ones, one can assume this approach to produce a number of benefits.

- Whereas acoustic features will result in large arrays of data, symbolic representations are much more compact, using just a small fraction of the original memory space.

- While computing alignments using DTW- or edit-distance-like algorithms, the number of frames per sequence can be dramatically reduced from a fixed ratio of frames per time unit to one frame each time a note onset or offset occurs.

- Using a transcription given in MIDI format, obvious errors of the feature extraction process can be recognized and handled prior to the actual alignment step. Examples of such obvious errors are detected chords or notes with pitches which do not occur within the score of the current piece. This pre-processing, however, might also eliminate incorrect notes played by the performer in certain cases.

An approximate transcription of an audio file can be directly based upon the Pitch Activation feature. The simplest method is to set the note boundaries to the times where the activation value h_i^p of a pitch i becomes greater than zero and falls back to zero again respectively. [Niedermayer, 2008] shows that almost all of the actually played notes (more than 99%) have an overlapping representation within such a symbolic representation. Also, not only does this exploit the sparseness of the factorization result, but an additional data reduction is performed since note velocities are set to a single value, such as the maximum or the third quartile, for example, and the actual time-varying activation pattern during a note's sustain time is dropped.

The effect of this method in terms of the amount of data which has to be stored and processed is shown on the example of the recording of Mozart's piano sonata k.279 (see Section 2.1.2). The resulting MIDI representation contains 6275 notes using less than 150 kB of memory. This is a little more than 7.5% of the space needed to store the Chroma vectors calculated at a time resolution of 50 frames per second. For the original acoustic representation of the factorization result this relation is even more drastic. The activation patterns of 58 pitches (i.e., the pitch range used in the sonata k.279) require 11MB of memory, which is more than 70 times the space needed for the symbolic version of the feature.

For the actual alignment of this representation to the score, one can choose between several approaches. The most important design decision is whether to process the data split into chunks of a fixed length at an appropriate overlap ratio, or to split the data into segments which can be variable in length but, on the other hand, have a constant pitch content. In doing so, each chunk starts and ends at either a note onset or a note offset. This has the advantage that actual events are aligned instead of arbitrary timestamps. On the other hand, the implicit weighting of notes with respect to their length is lost. While in the frame-by-frame approach, a bad alignment of a chord which is sustained for a relatively long time is valued by a cost which is not only proportional to the quality of the alignment, but also to the duration of the respective notes, this

relation is lost when only note on- and offsets are compared and no additional measures are taken.

4.3.2. Local Distances

Independent from the segmentation of the data into analysis windows, a local distance measure defining the cost of aligning two frames or, in the symbolic domain, two events to each other has to be found. In Section 4.1 the Euclidean and the cosine distance have been introduced as good dissimilarity measures for Chroma vectors. Although those metrics could be applied on Pitch Activation data as well, they are not the best choice for several reasons.

In the first place, the feature produces a different quantity of deletion (false negative) and insertion (false positive) errors. Especially in high pitch ranges the majority of errors is made up by spurious note detections. Therefore, the two types of errors should be treated differently.

Secondly, the STFT used here as the transform into the frequency domain divides the spectrum into linearly distributed frequency bins. On the other hand, musical notes follow a logarithmic frequency scale. Therefore, the deeper a tone, the closer in the spectrogram it is to its immediate neighbors. In addition, higher pitches also exhibit significant energy in the lower frequency bins making it even harder to reliably detect low notes. Therefore local distance calculation should accommodate this fact by relatively tolerant penalizing of missing low notes in the audio feature.

A simple distance measure that combines these ideas and has yielded good results during experimentation is

$$d(h^s, h^p) = \sum_{i=0}^{N-1} diff(h_i^s, h_i^p) \tag{4.15}$$

with

$$diff(h_i^s, h_i^p) = \begin{cases} h_i^s * \alpha & \text{if } h_i^p = 0 \\ h_i^p * \beta & \text{if } h_i^s = 0 \\ |h_i^s - h_i^p| & \text{else} \end{cases} \tag{4.16}$$

where h^s represents a feature vector taken from the score and h^p represent one feature vector extracted from the recorded performance. α and β are the weights for missing and spurious notes respectively. Throughout our work, 1.2 and 2.0 have proven to yield good results. Experiments have further shown that alignments can be improved by ignoring missing notes lower than a threshold around C3 (midi pitch 48) at all. Also, taking the square root of d turned out to be advantageous in combination with Dynamic Time Warping as explained in Section 4.1.

4.4. Onset Matching

In contrast to performing a rudimentary audio transcription, [Müller et al., 2004] describe a similar approach where instead of notes, only the onsets are extracted for each pitch and than matched to the score representation. This onset extraction is based on a multi-rate bank of elliptic filters and the short-time root-mean-square power of the filter output (see Section 3.3). Onset candidates are then extracted using the half-wave rectified power derivative as a detection function. Local thresholds are then applied on this function to select the significant peaks.

As a last pre-processing step, the onsets are organized into temporal bins. To this end, the time scale is partitioned into equally spaced segments of a certain length – the authors use a segment length of 50 ms. Segments in which no onset candidate is detected are then dropped. In analogy to this, the score is divided into score bins, i.e., sets of notes which are played concurrently in the score. For the audio recording as well as for the score, the final feature representation consists of a sequence of sets of notes with one uniform onset time per set. In contrast to the score features, where each note has the same weight, the audio features additionally contain the relative strength, i.e., a MIDI velocity estimate, for each onset.

For the actual alignment, a dynamic programming approach similar to Dynamic Time Warping is applied. However, instead of using acoustic features on a linear time scale, symbolic onset lists are matched according to a similarity model. This model takes into account that the audio processing step is expected to yield a number of spurious peaks in addition to the correct onset. On the other hand, notes can be omitted or played in a way such that they are not detected by the signal processing layer of the system. The conclusion drawn from these two observations is that the matching model should be tolerant against insertions and deletion.

[Müller et al., 2004] propose a matching score allowing for octave errors in the pitch specific onset detection. To this end, the match score of a single onset extracted from

the audio recording amounts to its respective MIDI velocity, if the reported pitch equals a corresponding pitch within the score bin, its first, or second harmonic; otherwise it is defined to be zero.

Using this matching score, the globally optimal match is obtained by recursively calculating the accumulated score matrix S of size $(N+1) \times (M+1)$ from the local scores $s(i,j)$ between the i^{th} score bin and the j^{th} segment of audio features, as

$$S_{i,j} = \max \begin{cases} S_{i,j-1} \\ S_{i-1,j} \\ S_{i-1,j-1} + s(i,j) \end{cases} \tag{4.17}$$

where $S_{0,0}$, $S_{i,0}$, and $S_{0,j}$ are defined to be 0 for all $i \in [1,N]$ and $j \in [1,M]$. Therefore, the maximum global score $S_{N,M}$ is the sum over all $s(i,j)$, where (i,j) is part of the matching path. This path and the according matches can, in contrast to the Dynamic Time Warping algorithm, be easily backtracked using the accumulator matrix S only. Since the value of an intermediary score $S_{i,j}$ is only increased if a diagonal step is taken, a comparison for equality of a cell of S with its possible predecessors will reveal the direction the matching path has taken at that point.

The method, as described so far, treats onset cues within the audio file as simultaneously if they occur within the same segment of 50 ms. To obtain the accurate timings of individual notes within such a segment, the highest peak of the respective section of the onset detection function is assumed to be the onset of a corresponding score note. Again, for each pitch, the first two harmonics are considered as well. If no corresponding peak within the onset detection function is found, the note is marked as unmatched.

While this method yields a note-level alignment exploiting the robustness of the dynamic programming approach, it suffers from a number of drawbacks. First, two notes at the interval of one octave can, very likely, not be distinguished. Since for each note, the first two harmonics are taken into account in addition to the fundamental frequency, playing two notes which share these partials within the same audio segment is an inherently ambiguous and therefore error-prone scenario.

Also, the partitioning of the audio recording into chunks of 50 ms seems to be problematic. As shown in Table 2.6, a timing deviation between notes of the same chord of more than those 50 ms is very common. In such cases, the onsets of the individual chord notes will be fragmented over two or more audio chunks. The consequence is that the tonality of the chord is represented in neither of the individual segments and this information is lost.

While the authors present an evaluation of the runtime behavior of their system, the accuracy is evaluated only qualitatively. Also, a semi-automatic anchor note detection is performed to prevent the alignment from drifting off too far.

4.5. Conclusion and Consequences for this Thesis

In this chapter, basic methods for the alignment between two feature sequences, calculated from a symbolic score and an audio representation of the same piece of music, were described. Dynamic Time Warping was presented as the default algorithm if the score is represented in terms of audio features as well (A2A). In contrast, building graphical models – most commonly Hidden Markov models – from the score gives the designer a chance to incorporate additional information into the calculation which would be lost when transforming the symbolic data into the audio domain (A2S). As a third option, the extraction of symbolic information from the audio material is presented (S2S). The advantage of this method is that the amount of data that needs to be processed during the alignment step is reduced significantly. In addition, there is no need for a perfect transcription of the audio signal, since the notes which are played are known from the score and are likely to be correctly selected from a reasonable set of candidates during the matching phase.

The main objective of this thesis is to develop an alignment system optimized towards accuracy. Therefore, the quasi-transcription approach had to be discarded. As reported in [Niedermayer, 2009b] and [Müller et al., 2004], it is an appropriate means of handling very long pieces of music, such as audio recordings containing whole sonatas without a segmentation into movements. The recording of Mozart's sonata k.284, as described in Section 2.1 has an overall length of more than 26 minutes, for example. By quasi-transcription, the compactness of the feature representation in our experimental system was improved by a factor of 70. The resulting time of computation was also reduced to about a tenth. However, due to the audio transcription step, which can be described as only rudimentary, the method cannot compete with other systems concerning accuracy of individual notes' or chords' estimated onsets. A median timing displacement of 205 ms and a 95^{th} percentile of 905 ms are not sufficient for applications such as musical performance analysis.

Concerning Dynamic Time Warping and Hidden Markov models, both approaches suffer from the same shortcoming: they do not treat individual chord notes separately. While the separate parameter extraction for single notes within a chord is not possible in DTW as a matter of principle, asynchronous note onsets can be represented by graphical models. A corresponding method is described in [Devaney and Ellis, 2009]. However, as argued in Section 4.2.4 the idea of modeling each chord transition by all possible

combinations of (i) a note of the previous chord sounding, (ii) a silence or transient stage, and (iii) the respective note of the next chord sounding, will most likely exceed the available computational capacities. For the transition between two chords of a degree of polyphony of n, 3^n states are required to model all possible alternatives. The highest degree of polyphony found within the Mozart sonatas, i.e., 8, would result in 6561 states for one single chord transition.

Based on these considerations, we propose to use a multi-pass approach for accurate Audio-to-Score Alignment (a detailed description of such a system will be given in Chapter 6). A computationally efficient initial alignment is performed first based upon which each note is refined individually. This design decision was made independently from the actual technique applied for the alignment step. The decoupling of the rough identification of notes within a tolerance window of a certain range and the extraction of an accurate onset estimate within this window allows for stage specific evaluation and optimization.

[Dannenberg and Hu, 2003] argue for the use of Dynamic Time Warping instead of Hidden Markov models because of the simplicity of this approach. On the one hand, DTW is a special case of HMMs. Cells of the dissimilarity matrix correspond to states and the alignment cost is indirect proportional to the output probability. The strength of HMMs is the possibility to model complex state transitions and to learn output probabilities directly from adequate training data. On the other hand, the advantage of DTW, is that it yields reasonable results without such a training stage or an explicitly designed model.

We are in line with this reasoning and consider DTW results to be "accurate enough" to serve as first pass alignments. In our experiments, an STFT window size of 1024 samples and a hop size of 512 frames turned out to yield good results. The largest absolute temporal displacement produced by this initial (first pass) alignment on our entire evaluation corpus was about 4.5 seconds. While this values seems to be large, the 95^{th} percentile of absolute errors of 363 ms shows that there is only a very small number of such outliers. Slightly more than 70% of all aligned note onsets deviate from their actual time by less than 50 ms and for almost 30% the error is less than 10 ms (see Appendix B, for details).

5. Alignment Optimization Techniques

Dynamic Time Warping was chosen as the method to be used in our final system for several reasons discussed in Section 4.5. In the following, enhancements to the DTW approach concerning the issues of computational costs, robustness, and accuracy will be described. In Section 5.4, we will discuss which of these are used in our final system, and how.

5.1. Optimization towards Computational Costs

The DTW algorithm is, as well as other dynamic programming methods, of asymptotic complexity $O(n^2)$ in time as well as in space. Therefore, for pieces of music longer than a certain time it is impossible to keep a reasonable time resolution of features and still compute a global alignment. While relatively simple approaches put global, static constraints on the search space for valid paths, more complex methods apply adaptive constraints based on the actual data.

5.1.1. Static global Constraints

Two straightforward methods which limit the search space for valid alignment paths globally are the *Sakoe-Chiba band* and the *Itakura parallelogram* ([Rabiner and Juang, 1993], [Sakoe and Chiba, 1978]). The Sakoe-Chiba band implicitly assumes that the two sequences which are aligned contain the same information in the sense that no insertion or deletion of significant length has taken place. Further requiring the tempo difference between the two feature sequences to be arbitrary but fixed, would result in an alignment path along the main diagonal of the dissimilarity matrix. The Sakoe-Chiba band is, then, a tolerance range of a constant horizontal and vertical offset T with respect to the main diagonal, i.e., the constant tempo change

assumption. This allows for local deviations, such as expressive tempo variations, but prohibits paths from modeling structural changes between score and performance.

A similar concept is the Itakura parallelogram, which, instead of considering the average tempo difference between two feature sequences, is based upon an assumption about the maximum tempo deviation factor. This factor corresponds to a maximum and minimum slope of the alignment path. Introducing these slopes as boundaries within the dissimilarity matrix, starting from the two end-points of the main diagonal results in a parallelogram within which the optimal path is searched for.

Both methods, constraining the search space to a band or a parallelogram respectively, can reduce the asymptotic complexity from $O(n^2)$ to $O(n)$ depending on the implementation. The simple approach of setting all alignment costs outside the allowed search space to infinity and leaving the actual alignment algorithm unchanged, only reduces the computational effort of the dissimilarity computation step. It is, therefore, more a restriction to prevent implausible alignments than to reduce the computational effort. Implementing path constraints in the cost minimization step, however, will not only result in a significant speed-up but also in major memory savings. The forward step of the DTW algorithm can then be calculated at a space complexity of $O(1)$.

5.1.2. Online Audio Alignment

A completely different approach to align sequences of arbitrary length without posing the problem of memory space, is to perform the alignment task online, i.e., by considering one audio frame at a time and deciding on onset times for the individual score notes shortly after their occurrence, allowing only a small latency. Online Audio-to-Score Alignment is also referred to as *Score Following* and allows for numerous applications, such as automatic accompaniment (see [Dannenberg, 1984], [Dannenberg and Raphael, 2006], [Davies, 2007], or [Raphael, 2009]) or an automatic page turner (see [Arzt et al., 2008]) for live performances. However, processing the audio data as a stream is a more difficult problem in comparison to offline algorithms, since the computer system cannot ”look into the future”. Therefore, note candidates identified within the audio stream cannot be compared to competing ones occurring shortly afterwards.

An online variant of Dynamic Time Warping was first proposed in [Dixon, 2005b] and then developed further in [Arzt et al., 2008]. It is based upon a (ring) buffer holding the alignments between 500 feature vectors computed from the audio as well as from the score representation. The algorithm then selects between advancing to the next score frame, to the next audio frame, or both, depending on the partial alignment paths

ending at the newest row or column within the buffer. If the path with lowest costs amongst these candidates ends at the newest audio frame, a new frame is read from the stream. The analogous procedure is performed for the score. In the case where the minimum cost is yielded by the path ending at the alignment of the latest audio data to the latest score frame, the algorithm advances in both directions.

To tackle the robustness issue, [Arzt et al., 2008] describe a *Backward-Forward Strategy*, where the alignment path over a short period is reconsidered, more effectively explaining the accumulated information. To this end, a smoothed backward path is calculated and followed for a certain number of steps. Then, starting at the respective point of this path, a new hypothesis about the forward path is obtained by applying the default dynamic programming approach. This new path, however, is expected to be more robust since it relies on more information which was unknown at the time the initial alignment was computed and is, therefore, used by the algorithm to continue. The authors suggest to alternate between short backtrackings of 10 steps and a longer one with a length of 50 steps, i.e., 200 ms and 1 second in their specific implementation.

Another strategy to overcome the drawback of not knowing the whole audio recording in advance, proposed in [Arzt et al., 2008], is to incorporate information about note onsets. The basic assumption is that onsets can be detected almost instantaneously by taking into account spectral novelty and rapid increases of the signal energy. Also, by following the score and maintaining a current tempo hypothesis the timing of the next note can be anticipated up to a certain accuracy. Combining those cues and the score information, the onset of a note can be detected without observing the "future" sustain phase of a note.

5.1.3. Path Pruning

Path pruning is a greedy approach where only promising paths, i.e., those paths with an overall cost below a certain threshold θ, are continued during the dynamic programming phase. [Soulez et al., 2003] describes a method where the accumulated path costs are calculated row by row. The threshold $\theta(m)$, which an element in the m^{th} row must not exceed to become a valid candidate for a continuation at the $(m+1)^{\mathrm{th}}$ row, is set to

$$\theta(m) = 1.1 \min_{n} [accu(m, n)] \tag{5.1}$$

To allow for more possible paths, those between the outermost candidates and the main diagonal are not pruned, resulting in a continuous corridor. The authors report a data reduction by a factor of 90, reducing a feature sequence of length 36000 (corresponding

to a piece of a length of 3 minutes at a hop size of 5.8 ms, i.e., 256 samples at a sampling rate of 44.1 kHz) to a corridor of an average length of 400.

In analogy to this method, instead of processing one row at each iteration, thresholds $\theta(l)$ for each length l of a path can be used to drop candidates which are already too expensive at an early stage.

However, neither variant of the Path Pruning approach guarantees that the globally optimal alignment path is found. Paths which are cheap, in terms of alignment cost, at their beginning and become increasingly expensive towards their end are expanded. On the other hand, a globally cheaper path might be discarded if it has to traverse a section where the feature sequences are relatively dissimilar at its beginning. To ensure that the global optimum is found, a tentative optimum is obtained by calculating the alignment cost along the main diagonal. Partial paths exceeding this value can be discontinued while the algorithm is still guaranteed to yield the global optimum. Although the tentative cost minimum can be updated as soon as cheaper alternatives to parts of the main diagonal are found, this method will result in a lower speed-up. Since the cost of a complete path is considered as a pruning criterion, paths are not likely to exceed this value during an early iteration.

5.1.4. Shortcut Paths

As an alternative to continuing only promising partial paths or restricting paths to those within a plausible search space, [Orio and Déchelle, 2001] propose to shorten the paths by considering only those score-frames where the tonality is changed, i.e., frames where a note onset or an offset occurs. The time as well as the space complexity is decreased from $O(n^2)$ to $O(nm)$, where m is the number of distinctive score events for which, in practice, $m \ll n$ holds.

Similar to this approach is the alignment in the symbolic domain, as described in Section 4.3. By performing a rudimentary transcription on the audio data, not only the score representation, but also the audio recording can be reduced to a sequence of note or chord events.

5.1.5. Multi-Scale DTW

Another approach reducing time and space complexity to $O(n)$ (or $O(n \log n)$, depending on the exit-condition) is multi-scale Dynamic Time Warping (see [Salvador and Chan, 2004], [Salvador and Chan, 2007], and [Müller et al., 2006], for

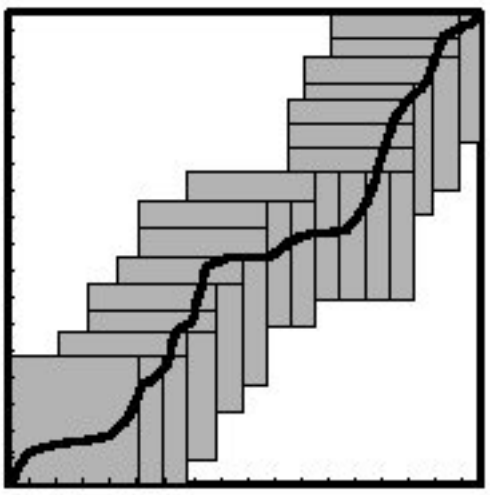


Figure 5.1.: Two-scale DTW: An approximate alignment path (black) through the dissimilarity matrix is calculated in the first pass. In the second pass, only matrix elements within a certain range around the initial path (gray) are considered. Since after the initialization only the respective last row and column must be kept in memory, the time resolution can be increased by a significant factor.

example). Here, an initial estimate of the optimal path is obtained at a low time resolution and then iteratively refined at increasing feature resolutions. At each of the refinement steps, paths are constrained to stay within a band determined by a certain tolerance range around the tentative optimum. As for Path Pruning, the result of a multi-scale DTW is not necessarily the global optimum. It may happen that low resolution features are misleading in such a strong way that the actual path yielding minimal costs is out of the search radius.

In [Niedermayer, 2009b], an implementation using only two steps is used. In the first one, a standard DTW is computed on features extracted from an STFT spectrogram at a window as well as a hop size of 4096 samples ($\sim$93 ms). This allows for the processing pieces of lengths up to more than 25 minutes, corresponding to a dissimilarity matrix of a size of about 2GB. The second step is the refinement, calculated on features based on a different spectrogram with the hop size reduced to 512 samples ($\sim$12 ms). As illustrated in Figure 5.1.5, the path estimation from the first step leads the search in the second step such that only dissimilarity values and path costs within an area of radius r frames need to be calculated. In this way memory requirements can be kept low by just storing dissimilarity measures of the currently processed element and path costs for the respective last rows and columns, as depicted in Figure 5.1.5.

5.1.6. Divide & Conquer

[Niedermayer and Widmer, 2010b] propose a divide and conquer approach to improve the efficiency of DTW. This idea was originally introduced by [Müller et al., 2004], where, however, the interaction of a system user was required. Given a set of anchor notes, for which the exact timing is known, solving the alignment problem on the whole piece can be reduced to finding optimal alignments between each pair of consecutive anchor notes. Given a maximal interval c between two anchors, the sub-DTWs are computed in $O(c^2)$ in time as well as in space. While the space complexity of $O(c^2)\widehat{=}O(1)$ does not change compared to considering the whole piece, time complexity decreases to $O(c^2 * N/c) = O(c * N)\widehat{=}O(N)$, while guaranteeing that the globally optimal alignment is found. This is a great improvement over the original algorithm's complexity of $O(N * M)$, where N and M are the lengths of the audio and score feature sequences respectively.

This increase in efficiency is countered by the additional problem of how to identify suitable anchor notes and how to extract their respective onset times. [Müller et al., 2004] describe an approach where the user manually selects an anchor configuration or verifies suggestions made by the algorithm. These suggestions are established based on cues such as pauses, long isolated fortissimo chords, or notes with salient fundamental pitches, i.e. pitches that do not overlap with harmonics of concurrently played notes.

In [Niedermayer and Widmer, 2010b], this idea was developed further in order to avoid the requirement for user interaction. Based on an initial, low resolution alignment, note onsets are refined using the Pitch Activation feature. This feature is calculated from the spectrogram within a certain search window around the initial onset estimate, while exploiting the knowledge about the notes which are expected to sound within this window. A significant increase in the activation energy is then found to be an accurate onset estimate. While such a refined onset is obtained for each note, only those notes where the confidence level concerning correctness and accuracy issues exceeds a threshold are selected as anchor notes. The exact procedure is described in detail in Section 6.2.

5.2. Optimization towards Robustness

In Section 3.2, the Chroma feature was introduced as a fairly robust descriptor of the tonality of a frame. This generally holds for different dynamic ranges, timbres, or playing styles. However, in specific situations, such as sections of a very high degree of polyphony, when glissandi are played, or when heavy distortion is used as an audio

effect, the Chroma feature loses its explanatory power, since all pitch classes have an almost equal presence. On the other hand, a highly repetitive structure of a piece or errors made by the performer can mislead the alignment algorithm to such an extent that it can be considered to be failed.

In the literature, several methods can be found which either aim at enhancing alignment robustness by introducing additional types of features and subroutines which handle common problems, such as structural deviations of the performance from the score of a piece, or at automatically obtaining a confidence measure reflecting the plausibility of an alignment.

5.2.1. Short-Time Statistics

In the context of an alignment-based audio retrieval task, [Müller et al., 2005] propose the usage of short-time statistics over Chroma features to increase robustness towards different articulations and local tempo changes. To this end, the Chroma vectors are normalized to a sum of 1 before quantizing the individual values v_i according to

$$\hat{v_i} = \begin{cases} 4 & \text{, if } v_i \geq 0.4 \\ 3 & \text{, if } 0.2 \leq v_i < 0.4 \\ 2 & \text{, if } 0.1 \leq v_i < 0.2 \\ 1 & \text{, if } 0.05 \leq v_i < 0.1 \\ 0 & \text{, otherwise} \end{cases} \tag{5.2}$$

The sequence of quantized feature vectors is then convolved element-wise using a Hann window. In their specific implementation, the authors use a window of a length of 41 frames which corresponds to 4.1 seconds. The result is downsampled to a feature resolution of 1 Hz and the respective feature vectors are normalized to a Euclidean norm of 1. The authors call this feature *Chroma Energy distribution Normalized Statistics* (CENS).

Its main strength over the Chroma feature is that, by considering short-time statistics instead of single feature values, local variations in timing and articulation are smoothed out to a large extent. The same holds for noise caused by transient spectral changes at the note onsets. Noise which is inherently present in Chroma representations due to harmonics contributing to pitch classes different from the one corresponding to the fundamental frequency is also suppressed by the thresholding in Equation 5.2. This

straightforward detection of noisy contributions would not be possible if the longer analysis window were already applied at the Chroma calculation stage.

5.2.2. Robustness to Structural Changes

A well known problem in Audio-to-Score Alignment are changes of a piece's structure between score and performance. This can be due to individual variations of a performer, who can intentionally leave out a repetition or introduce one that is not notated in the score. On the other hand, OMR software, used to digitize a score, might have missed a repeat mark or a score MIDI file does not contain redundant parts to save memory. Independent from the cause of structural changes, they bear the risk that an alignment will fail. Although Dynamic Time Warping yields the optimal alignment, when one feature sequence contains an additional section, an alignment path might be cheaper if it successively proceeds in both directions at an adequate slope than if it remains in a "waiting" position within one representation until the inserted block within the other feature sequence is completed.

As a solution to this problem, [Arzt et al., 2008] use a system which keeps track of multiple path hypotheses at a time. To this end, the score representation is assumed to contain information about the major sections of a piece and respective points where a performer can continue after changing the structure. Starting from these points, several potential paths are computed in parallel. After a certain number of audio frames, these candidates are evaluated based on the overall path cost and the cheapest one is selected for continuation. To avoid excessive computational efforts, the authors propose to only consider three path candidates each time the performance reaches a section boundary. One, assuming that the performer repeats the played section, a second one, where the performer continues playing the subsequent section, and a third path hypothesis, based on the assumption that the entire next section is skipped.

This approach was further refined in [Arzt and Widmer, 2010] to drop the requirement for annotated sections as part of the score representation. Instead, a rough position estimator was introduced which works on feature sequences at a low time resolution corresponding to a hop size of 300 ms. Based on this data, multiple promising points in score time are selected and the respective paths are refined at a higher time resolution, before a final decision on the actual most likely alignment is made.

Instead of using multiple matchers, [Fremerey et al., 2010] formulate an extended Dynamic Time Warping algorithm in the context of offline alignment, already incorporating the possibility to jump between sections of a piece. Again, the section boundaries are supposed to be known from a prior analysis of the score by means of OMR software

or by human annotation. Then, in order to account for structural changes, the monotonicity constraint of DTW is dropped. To this end, in addition to the regular steps within an alignment path (see Equation 4.3), a jump to the beginning of an arbitrary section is introduced. To prohibit jumps which are not musically motivated, this option is only allowed (i.e., charged with finite costs) when the score is at a position of a jump directive.

Considering *da capo al fine* notations, it also makes sense to relax the end-point constraint of Dynamic Time Warping. Instead of ending at the last frame of the score representation, an alignment path is allowed to end at the last frame of a "da capo" section.

5.2.3. Plausibility Estimation

Although the estimation of the plausibility of an obtained alignment does not improve the robustness itself, it is a valuable means of identifying failed alignments as a first counter measure. [Turetsky and Ellis, 2003] describe the automatic extraction of "ground-truth" annotations of audio recordings by means of Audio-to-Score Alignment, which can then be used for the evaluation of a task with less information input. To recognize pieces where the offline alignment has failed, five plausibility measure are introduced: *Average Best Path Score*, *Average Best Path Percentile*, *Off-Diagonal Ratio*, *Square Ratio*, and *Line Ratio*. They can be grouped into methods inspecting the costs along the alignment path in relation to the "off-path" values within the dissimilarity matrix (the first three measures), and ones aiming at the shape of the optimal path itself (the latter two measures). In the following we will describe the Average Best Path Cost, our adaption of the Average Best Path Percentile, and a new measure we propose for the detection of paths with implausible shapes.

Average (Best) Path Cost

Given the alignment, an intuitive measure for the quality of the match between two pieces of music would be the average cost along the alignment path. For this calculation, [Turetsky and Ellis, 2003] use a median filter and consider values along diagonal regions of the path only. However, preliminary experiments have shown that this measure is too simple. Although Chroma vectors as well as Pitch Activation are relatively robust to changes in instrumentation or accompaniment as well as room acoustics or audio effects, higher average alignment costs can still be caused by such variations.

Relative Path Cost

One approach to account for differences in instrumentation or accompaniment is to calculate the average cost along the alignment path in relation to the overall average cost $\overline{D}$ over all D_{ij}, as described in [Niedermayer et al., 2011b]. The cost $\overline{D}$ is a good baseline, estimating the average path cost of a random alignment. It accounts for all specifics of the two feature sequences under consideration which the used feature is not entirely invariant to (e.g., changed recording conditions, varying levels of noise, or different arrangements). A similar measure – the Average Best Path Percentile – is based on the percentile of all path costs within all values of D. It was proposed by [Turetsky and Ellis, 2003] and shown to outperform the other measure mentioned above.

Off-Diagonal Cost

We have introduced the *Off-Diagonal cost* in [Niedermayer et al., 2011b] as a measure for the plausibility of an alignment path's shape. Despite its similar name, it is not related to the *Off-Diagonal ratio* of [Turetsky and Ellis, 2003] which takes the values along the path into account.

An inherent property of the DTW algorithm is that it is robust to small deviations of one sequence compared to another one. This is necessary in order to compensate for expressive variations or playing errors. However, the DTW algorithm's flexibility can also result in relatively low alignment costs when two different but still similar melodies are compared to each other. Especially in cases where the performance contains a lot of additional ornamentations, the desired result can have a similar alignment cost as is obtained by partly deleting the actual ornamented notes and matching the auxiliary notes to the score of a different melody.

This undesired behavior is likely to be detected by investigating the shape of the alignment path. Matching melodies, although with varying ornamentations, are likely to result in approximately linear paths along the main diagonal. In contract, relatively low global alignment costs observed while comparing a score with a performance of a different piece is likely to be achieved by major stretches and compressions of notes. This corresponds to significant horizontal or vertical segments within the alignment path. To measure this effect, we define the *Off-Diagonal cost* as the deviation of the optimal alignment path from the best strictly linear alignment path.

A method to retrieve linear segments, known from the field of image processing, is the Hough transform (see [Princen et al., 1992] or [Atiquzzaman, 1994], for example). It

transforms points $(i, j)^T$ from the image domain – in the case of audio alignment, the element-wise inverse of the dissimilarity matrix D, i.e., the similarity matrix S[1] – into the Hough space H, where each point $(\rho, \theta)^T$ represents a line given by θ – the angle with respect to the image domain's positive x-axis – and ρ – the distance from the origin, such that

$$\rho - i\cos\theta - j\sin\theta = 0 \tag{5.3}$$

A single point $(i, j)^T$ of the image domain lies on infinitely many lines at arbitrary angles θ. Thus, a point's Hough transform is a function $r_{i,j}$, following from Equation 5.3 as

$$r_{i,j}(\theta) = i\cos\theta + j\sin\theta \tag{5.4}$$

$r_{i,j}$ is a sinusoid of period 2π having a magnitude of $|(i, j)^T|$ and a phase of $\arctan j/i$.

In the discrete case the Hough space is sampled and represented by an accumulator array $\hat{H}$ of size $T \times R$. The function $r_{i,j}$ then becomes a set of corresponding cells, defined as

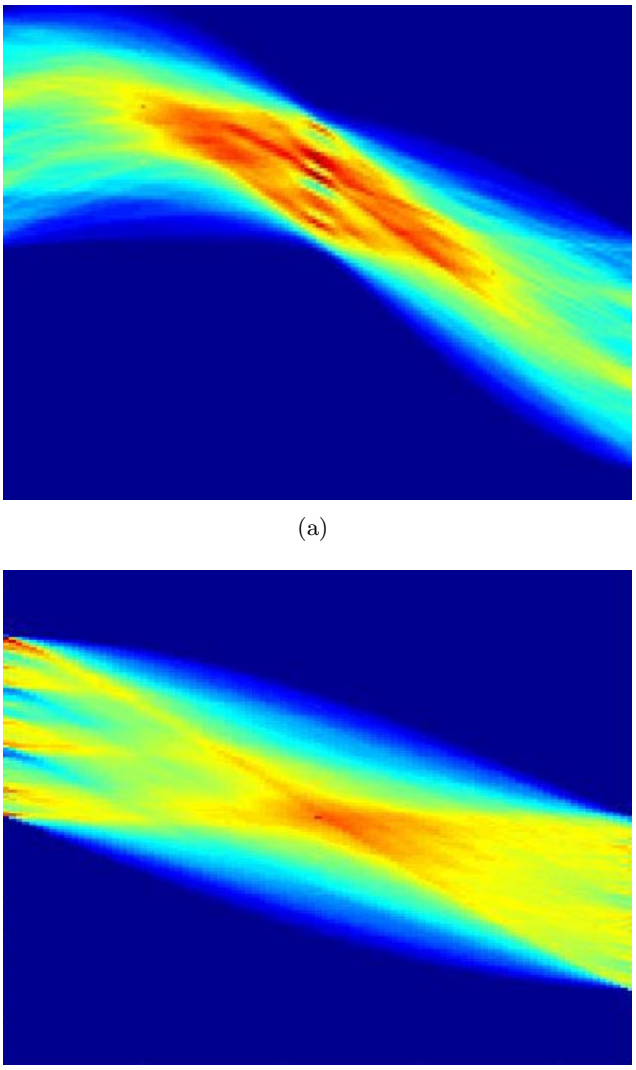
$$R_{i,j} = \{\hat{H}_{t,P^{-1}(h_{i,j}(\Theta(t)))} : t \in [0, T-1]\} \tag{5.5}$$

where $\Theta(t)$ is the angle θ corresponding to index t and $P^{-1}(\rho)$ is the sampling index ρ resolves to.

When applying the Hough transform to the similarity matrix S, for each element $(i, j)^T$, all accumulator cells in $R_{i,j}$ are increased by $S_{i,j}$. High values within the resulting $\hat{H}$ indicate prominent lines in the image domain. In Figure 5.2 the accumulator arrays $\hat{H}$ are shown that result from comparing two feature sequences calculated from performances of the same piece of music and and two independent recordings, respectively.

In our proposed system for the identification of versions of a same piece (see Section 7.4), the size of the accumulator array was set to the size of the similarity matrix S. Doing so yields fine resolutions if S is large and coarser resolutions if the compared features sequences are short. In addition, the angle θ was restricted to have a maximum

[1] Remark: Probable lines are represented as local maxima in the Hough space, which can be seen as an accumulator for evidence for lines described by certain parameters. Since this can be considered an "inverse" problem to cost minimization, a similarity measure is required instead of a cost, i.e., a dissimilarity measure



(a)

(b)

Figure 5.2.: Hough transform of two feature sequences calculated from performances of the same piece of music (a) and the recordings of two pieces different from each other (b).

deviation from the main diagonal of $\pm 30°$. Since the dominant line is assumed to be the best linear alignment path, this restricts the valid slopes – i.e., the tempo deviations – to reasonable values.

In summary, the Hough transform is used as a line detector. Applied on the inverse alignment costs, it finds linear segments within the similarity matrix S along which the two feature sequences under consideration match relatively well. Such alignments allow only for an offset between the first note onsets within the two musical representations and a constant tempo change. The highest value of the accumulator array $\hat{H}$ represents the best alignment under these constraints.

Finally the Off-Diagonal cost is calculated from the sets of points $A = \{a_{ij} \in \text{path}_{DTW}\}$ and $B = \{b_{kl} \in \text{path}_{linear}\}$. From the set A the points corresponding to the first and the last 5% of the path are discarded, to not penalize different offsets. For the remaining points a_{ij} along the path obtained by the DTW algorithm the cost $c(a)$ is calculated as the minimum of the horizontal and the vertical offset to the linear path.

$$c(a_{ij}) = \underset{b}{\text{argmin}}\ dist(a_{ij}, b_{kl}) \tag{5.6}$$

with

$$dist(a_{ij}, b_{kl}) = \begin{cases} |i-k| & \text{, if } j = l \\ |j-l| & \text{, if } i = k \text{ and } j \neq l \\ \infty & \text{, else} \end{cases} \tag{5.7}$$

From these individual values, the final Off-Diagonal cost c_{od} is computed as

$$c_{od}(A) = \frac{1}{|A|} \sum_{a \in A} c(a)^2 \tag{5.8}$$

As also described in [Niedermayer et al., 2011b], this plausibility measure was used for the purpose of version detection, when the audio recordings of two performances of a same piece are aligned to each other. In the context of this work, the Off-Diagonal cost was calculated on alignments between short chunks of the two respective feature sequences. In this way, similar chunks are detected without raising the issue of structural changes (see Section 7.4).

Other Plausibility Measures

A method, similar to the computation of the plausibility of an alignment path over short chunks, is to detect linear segments within the complete alignment path. [Turetsky and Ellis, 2003] define the *Line Ratio* as the length of approximately linear alignment path segments in relation to the overall path length and the *Square Ratio* as the area of rectangles around the linear path segments compared to the overall area of the dissimilarity matrix D.

The *Off-Diagonal ratio* introduced by the same authors is, despite its similar name, not related to the Off-Diagonal cost as defined above. It is generally not a measure aiming at the shape of an alignment path, but at its average cost in comparison to the costs of a path with an offset of a small number of frames. Thus, it is an indicator for a sharp peak around the cost minimum obtained by the DTW algorithm.

5.3. Optimization towards Accuracy

Alignment accuracy is the main objective of this thesis. Therefore, we describe our proposed system including novel refinement strategies in a separate chapter (see Chapter 6). The optimization methods reviewed in this section describe related systems. Some respective approaches have already been discussed in previous sections within another context where accuracy was not the main issue and are here summarized as *implicit accuracy improvements*. In addition, methods which explicitly address the problem of resolving individual chord notes are explained.

5.3.1. Implicit Accuracy Improvement

A method which was introduced in Section 5.1.5 as a means to reduce the computational complexity of the DTW algorithm from $O(n^2)$ to $O(n)$ is multi-scale Dynamic Time Warping. While this approach can be used to compute an alignment at a fixed feature time resolution with less effort compared to standard DTW, one can as well argue that assuming a fixed computational capacity, alignments can be obtained at a much higher temporal resolution. From this point of view, multi-scale DTW can improve the accuracy of an Audio-to-Score Alignment.

[Ewert and Müller, 2009] use onset-based features, as described in Section 3.3, to yield accurate alignments. To not trade accuracy for robustness, identified onset candidates are summarized in a Chroma representation. To this end, the audio signal is transformed into the spectral domain using a multi-rate filter-bank, as described in [Müller et al., 2004], and the short-time root-mean-square power. Onset candidates, characterized by significant energy increases, are extracted and subjected to a peak picking mechanism. The remaining candidates are integrated over a longer analysis window and summed up according to the pitch classes of their respective pitch. After a normalization step, the sparse onset descriptors are expanded by adding a temporal decay. Each onset candidate is copied to a fixed number of subsequent frames and accordingly multiplied by a decreasing weight. The authors call the resulting representation *Decaying Locally Adaptive Normalized Chroma Onset* feature. Its strength

is that it contains accurate onset information while offering the robustness of Chroma vectors. Nevertheless, [Ewert and Müller, 2009] report best results to be obtained by using a combination of this onset-based feature and straightforward Chroma vectors in the sense that the respective dissimilarity matrices are simply added during the DTW calculation.

Although the described methods improve the accuracy of an Audio-to-Score alignment in general, they do not resolve the problem arising from asynchronies as contained in a natural performance. In Section 2.4.2, the importance of considering micro-timing in an Audio-to-Score Alignment system was shown. Chords where the individual notes' onset times spread over several hundred milliseconds are not an exception. These timing deviations are even more dramatic when also considering notes marked as grace notes or ornamentations in general. In such cases – which might not be apparent from score representations such as the MIDI file format – note onsets deviate from each other by up to one second, although they are notated concurrently in the score.

5.3.2. Score-guided Audio Transcription

Audio Transcription is the task of obtaining a symbolic representation, such as a MIDI file, from an audio recording and therefore a combination of Onset Detection, Offset Detection, and Multiple-Pitch Tracking. A small number of systems, such as the one presented in [Böck and Widmer, 2012], additionally extract the relative loudness of each individual note. Score-guided transcription can be considered an optimization towards accuracy since it is a method to obtain onset times at the note-level, i.e., distinct onset times for individual chord notes. Therefore, systems following this approach are strongly related to the focus of this thesis. In this section, after a brief introduction to audio transcription, the score-guided transcription system of [Scheirer, 1997] will be discussed.

A review of transcription methods is presented in [Hainsworth, 2001] and [Hainsworth, 2003], clustering them into three main approaches. The first systems were built on pure bottom-up principles, without considering any higher level knowledge. Although these algorithms used to fit very specific cases only, works like [Klapuri, 1999] or [Plumbley et al., 2006] show that bottom-up methods have overcome those early restrictions. A second group of transcription methods, like the ones used by [Martin, 1996], is based on blackboard systems. Here, low-level information gathered by digital signal processing and frame-wise description of the auditory scene as well as high-level prior knowledge is used to support or discard hypotheses at multiple levels. The third major approach to music transcription is made up by model based algorithms. Similar to blackboard systems, they also include high-level information

as well as low-level signal based features. The difference is that prior knowledge is fed into the system by introducing a model of the analyzed data. The signal is then processed in order to estimate the model's parameters. The results of these methods can only be as good as the degree to which the model can fit the data. Works like [Ryynänen and Klapuri, 2005] or [Godsill and Davy, 2002] are examples of this class. In a wider sense, neglecting the fact that the system was given knowledge about the score, the graphical model based Audio-to-Score Alignment approach could be related to the latter class as well.

In [Niedermayer, 2008], we have proposed a transcription system based upon the Pitch Activation features as described in Section 3.4. The usage of tone models to decompose the spectrogram was inspired by insights into transcription strategies of trained musicians. They have been shown to use a large amount of background information, such as the style of the piece or the instruments playing (see [Hainsworth, 2003]). In analogy to a human annotator, who would not listen for distorted guitar sounds in the recording of a piano piece, specific piano tone models are used during the extraction of the Pitch Activation feature.

We showed that simply setting note boundaries at timestamps where the activation level of a certain pitch becomes greater than zero and the time where it falls back to zero, yields a note-level overlap of more than 99%. This means that almost all notes which are played during the performance, overlap with a transcribed note at least to some extent. However, due to the significant number of spurious note events, the precision of this representation was only around 20%, making it useless as a meaningful transcription. To enhance the precision and the respective f-measure, a classifier was trained to distinguish between actual notes and false positives. In this way, the f-measure was improved from 0.36 to 0.92 at the note level. At the frame level an f-value of 0.52 was achieved in a 10-fold cross-validation experiment.

An early system which uses such an audio transcription approach instead of Dynamic Time Warping or graphical models, was presented in [Scheirer, 1997]. In this work, in each iteration, the algorithm searches for offsets or current amplitudes of those notes which have been detected before. Then, the onset of the next note within the score is extracted. Using this information, the score is reconsidered and the tempo estimation as well as the predicted onset times of the next notes are updated.

Since this method works online, i.e., processes the audio data as a stream without taking the whole data into account, the correct detection of the note onsets is crucial. Although not evaluated by the authors, it is reasonable to assume that this approach is less robust to playing errors or very noisy passages than an alignment-based system. Once a note onset is identified, there is no indication for a reconsideration of this

decision, such as the *Backward-Forward* strategy introduced in [Arzt et al., 2008] and described in Section 5.1.2.

The onset detector the system proposed by [Scheirer, 1997] in based upon takes into consideration several cues. First, the approximate timing of the next note is determined based on the current tempo estimation. Then, if, according to the score, only one note is struck at this time, the algorithm searches for high frequency energy content caused by the piano's hammer strike and also an increase in the overall spectral power. A note onset is reported when at least one of these two conditions holds while a positive derivative can be observed in the bin corresponding to the fundamental frequency of the pitch under consideration. If no significant peaks are found, the signal is filtered using a comb filter tuned to the fundamental and harmonic frequencies of the respective pitch. The onset is then detected, based upon the sharpest increase in energy of the filtered signal, as the point in time where the energy derivative becomes positive before attaining this steepest slope.

In cases where, according to the score, not a single note but a chord is played at a time, taking into account the high-frequency content of the signal, which results from the transient note onset, is ambiguous. While an energy increase in this region is a good indicator for an onset in general, it does not discriminate between the pitches that have been played. [Scheirer, 1997] proposes to isolate the onset of an individual note within a chord using multi-bandpass filters. The filters are designed to have pass bands around the fundamental frequency of the note and those harmonics which are not shared with another currently played pitch.

[Scheirer, 1997] also presents an evaluation of this approach. The used corpus of test data consists of three scales as well as five performance excerpts from pieces by Bach and Schumann recorded on a Yamaha Disklavier. The reported standard deviation of the displacements between extracted and actual timing is reported to be between 10 ms and 116 ms, depending on the complexity of the audio material. There is a performance with a standard deviation of above 100 ms. This is an outlier, where the transcriptions system failed due to a tempo change which is more dramatic than assumed by the internal tempo tracker. However, since significant tempo changes are an inherent characteristic of classical music, one can conclude from these results that relying on narrow tempo hypotheses compromises the robustness of the system. Although results reported by different authors cannot be compared directly, the temporal accuracy of Scheirer's system seems to be in the range of what DTW based approaches can achieve.

Two similar systems are the ones proposed by [Müller et al., 2004] and [Niedermayer, 2009b] (cf. Section 4.3), with the main difference that there are two distinct passes for obtaining a high-level description of the audio recording and

for the actual alignment. While [Müller et al., 2004] did not perform a quantitative evaluation, the accuracy reported in [Niedermayer, 2009b] is lower than what we achieve by simple DTW (cf. Appendix B). This strengthens our arguments in favor of a multi-pass system for accurate alignment.

5.3.3. Single-Pass Post-processing Methods

A number of authors have developed post-processing methods, where the onset of each individual note is refined within a search window around a tentative onset estimate obtained by an initial alignment. In doing so, the advantages of the two approaches described above – the accurate extraction of individual notes' onset times and the robustness of dynamic programming – are combined. This approach is preferred over alternatives, such as modeling asynchronies by means of hierarchical HMMs where each note or chord transition is represented by a number of sub-states according to all possible successions of note offsets and onsets, due to its flexibility and computational efficiency.

In the following, two approaches will be reviewed. The first one is based on an audio transcription technique – pitch specific note onset detection. There, an initial alignment based on Dynamic Time Warping is performed, before the onset of each note is reconsidered individually. The second approach is based on machine learning. The initial alignment is used as training data for an onset classifier. After finishing the training, the classifier is applied to the same piece of music, yielding more accurate onset estimates which are then used to iteratively retrain the classifier. The method we will introduce in our system (Chapter 6) differs from these methods by using multiple post-processing methods where note onsets are further refined if the level of confidence into the estimate from the first pass is not high enough.

Pitch-specific Onset Detection

While onset based audio features, as, for example, applied in [Ewert and Müller, 2009] as a strategy to refine audio alignments, are an integral part of the feature extraction and core alignment phase of a system, [Meron and Hirose, 2001] introduce a post-processing approach to refine Audio-to-Score alignments. The proposed system first computes an initial alignment using DTW. Then, for each individual note, a more accurate onset is searched for within a window of predefined size around the tentative onset estimate.

To avoid confusions due to overlapping partials between two notes which sound simultaneously, those harmonics which can unambiguously be related to a certain pitch are given more weight. This is similar to the idea presented in [Scheirer, 1997] and described in Section 5.3.2, however differing in so far as overlapping partials are not generally neglected but only down-weighted. The basic energy of the h^{th} harmonic of a pitch with a fundamental frequency of f_0 is computed in the time domain, as

$$e_h(l) = \left[\sum_{k=1}^{K} s_l(k) \cos\left(k\,(h+1)\,f_0\right)\right]^2 + \left[\sum_{k=1}^{K} s_l(k) \sin\left(k\,(h+1)\,f_0\right)\right]^2 \tag{5.9}$$

where s is the l^{th} segment of length K of the signal. The overall energy of a note i at the frame s_l is then given by

$$E(i,l) = \sum_{h=0}^{H} w(i,h)\, e_h(l) \tag{5.10}$$

The weighting function $w(i,h)$ does not only take into account the pitch of note i and its respective h^{th} harmonic with frequency f_h, but also the score time. Amongst all notes which are expected to sound simultaneously according to the score, the partial with the frequency closest to f_h is obtained. The resulting weight is then proportional to this difference such that those partials where a confusion is least likely yield the highest weights.

The authors report a reduction of the standard deviation of timing errors from 43 ms down to 25 ms due to this post-processing step. However, they manually initialize the DTW alignment. While listening to the audio recording, a human annotator presses a key at each beginning of a word of the lyrics. In this way, the computed alignment can be forced to only deviate from this reference within a certain tolerance range. A very accurate first alignment then allows for the use of relatively narrow search windows in the refinement step. One can assume that a fully automatic system would produce some outliers and therefore a larger standard deviation of alignment errors.

Onset Classification

[Hu and Dannenberg, 2005], [Hu and Dannenberg, 2006], and [Liu et al., 2010] describe a different refinement method based on an initial DTW alignment. There, the initial alignment is used as training data for an automatic onset classifier. The classifier

is initialized using audio material synthesized from a known score. Then, the training on an arbitrary audio recording is performed as follows.

1. Perform an onset detection on the audio recording resulting in an classifier output function $v(t)$ indicating the likelihood of the t^{th} audio frame to contain a note onset.

2. Compute the function $w(t)$ as a pseudo-probability distribution resulting from the note onsets as yielded by the Audio-to-Score Alignment. To this end, Gaussian windows, with the standard deviation σ being half the length of an audio frame, are centered around the predicted onset times and their values are summed up to obtain the function $w(t)$. In a last step, all values $w(t) = 0$ are set to a small number ϵ to also allow onset to occur at such times t at a respective probability.

3. Combine the classifier output v and the prediction obtained from the audio alignment into a new function v_{new}, defined as $v_{new}(t) = v(t)\, w(t)$.

4. For each note, find the maximum value of v_{new} within a search window around the tentative note onset as estimated by the DTW algorithm. The windows length to each side is not longer than the analysis window used during the alignment step and also bounded by the half of the inter-onset interval with regard to the adjacent note.

5. Retrain the classifier based on this data, where each estimated onset has a target value of 1 while all other frames are assigned a target value of 0, using a default error back-propagation method. The misclassification penalty is set such that the low number of onsets compared to non-onsets is compensated for.

The procedure is repeated until a local minimum of errors is reached in a 5-fold cross-validation. The classifier used is a multi-layer perceptron designed as a feed-forward neural network with two hidden layers which are fully connected. The first layer consists of 6 neurons while the second layer has only 4 neurons. The output is a real value $v \in [0, 1]$, with $v \geq 0.5$ indicating an onset.

Given an audio recording of a performance where the ground-truth transcription is not known, one can only use unsupervised machine learning techniques. However, due to the generally better understanding and the variety of well-studies algorithms, supervised learning approaches are preferred and assumed to yield better results. On the other hand, such techniques require a large number of training samples, i.e., annotated audio recordings, in this context. The advantage of the described *bootstrap learning* is that

it automatically acquires its own training data and improves the data quality in terms of onset accuracy with each iteration.

The features, i.e., values of the perceptron's input units, used in [Hu and Dannenberg, 2006] are the logarithmic energy of the frame, the frequencies of presumable partials, the energy of the first three harmonics of a pitch, as identified from the set of partials, in relation to the frames sum of energies, the inharmonicity factor, the zero-crossing rate, and the derivatives of these values. In contrast to the alignment step, where a window length of 50 ms is suggested, the features fed into the classifier are computed at window size of only 5.8 ms, i.e., 256 samples at a sampling frequency of 44.1 kHz.

5.4. Conclusion and Consequences for this Thesis

In this chapter, we presented several methods for the optimization of DTW-based audio alignment. Since the results of a default alignment using Chroma vectors show only a small number of outliers on our evaluation corpus, increasing the robustness is not a main objective here. A 95^{th} percentile of absolute time displacements of 363 ms is promising, so that we concentrated on accuracy. Nevertheless, a system for the automatic detection of versions of a same piece of music is described in Section 7.4 which relies on audio alignment and two of the discussed plausibility measures.

Although it has no direct influence on Audio-to-Score Alignment results, computational efficiency is an important issue. In our implementation we use a two-scale DTW, where an initial alignment is computed at a hop size of 4096 samples. In addition, an Itakura parallelogram is applied, allowing only for a certain maximum offset between the two feature sequences and an alignment path slope, i.e., tempo difference between the two respective sequences, within a defined band. Based on this initial alignment, a second pass at an increased temporal resolution is performed where only potential paths within a corridor around the first estimate are considered. There, a hop size of 512 samples and a search radius of 350 frames were found to be an adequate compromise between allowing for the correction of errors made at the first alignment and keeping the computational costs low.

Another strategy used in our Audio-to-Score Alignment system is the Divide & Conquer approach. In the refinement step anchor notes, for which an accurate onset time can be extracted at a high level of confidence, are determined automatically. Other notes which are notated concurrent to anchors or in between two such anchors are then reconsidered, taking this additional information into account.

An optimization technique which is not used in our final system are shortcut paths. However, they are the principle behind the increased computational efficiency of our alignment system working in the symbolic domain as described in Section 4.3.

Since accuracy is the main issue in this thesis, we present our final system for accurate Audio-to-Score Alignment in the next Chapter in detail. The methods described above are related work. The main difference of our proposed system is that is uses a multi-pass approach where notes which are relatively "easy" to detect constrain the extraction of note onsets which are less clear.

6. A System for Accurate Audio-to-Score Alignment at the Note Level

In this chapter, we describe our new system for the accurate extraction of each individual note's onset from an audio recording by means of Audio-to-Score Alignment. From a practical point of view, the objective is to minimize the number of notes for which a human annotator would need to correct the timing information.

In [Niedermayer and Widmer, 2010a] and [Niedermayer and Widmer, 2010b], we have proposed a number of post-processing methods based on an initial alignment obtained by Dynamic Time Warping to achieve Audio-to-Score Alignment at the note-level. In contrast to the single-pass approaches, the proposed post-processing chain works in several steps, starting with the refinement of notes where an onset candidate can be obtained at a high level of confidence. Such notes are called *anchor notes*, since they constitute fixed matchings that further enhancement steps rely on. The use of anchor notes has already been mentioned in Section 5.1.6, where the Divide&Conquer approach to reduce computational costs was discussed. In a similar manner, the piece is divided into relatively short sections of which the audio-score matches at the boundaries are known. Using this additional knowledge, notes concurrent to an anchor and those in between two such fixed points are revisited in two separate steps. An overview of this system is shown in Figure 6.1.

6.1. Initial Alignment

In an initial alignment step Dynamic Time Warping of sequences of Chroma vectors is used. The choice in favor of Chroma vectors instead of onset based features has been made due to the Chroma vectors' robustness to large numbers of inserted notes. Although onset based features yield more accurate results in an DTW-based alignment, they fail at certain pieces where up to 39 additional notes are inserted into one single

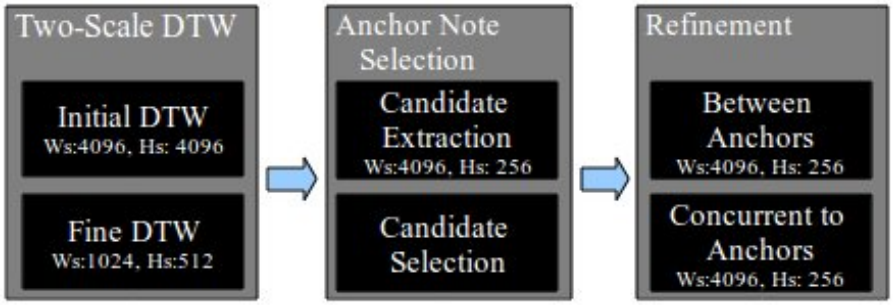


Figure 6.1.: Overview of the proposed Audio-to-Score Alignment system (*Ws* and *Hs* indicate the window and the hop sizes in samples chosen for the time-frequency transforms at the respective computations.)

measure. To achieve relatively accurate alignments using Chroma vectors, we have chosen a relatively high temporal resolution. In our implementation a hop size of 512 samples and a window size of 1024 samples are used.

To keep computational costs low, this alignment is computed using a two-scale approach. A first alignment path is computed at a coarser temporal resolution, i.e., a hop size as well as a window size of 4096 samples. The alignment at the finer temporal resolution is then computed only for possible paths within a search radius of 350 frames around the initial path estimate. To further speed up the computation, an Itakura parallelogram is used, allowing for maximum offsets between the features sequences of 100 frames, i.e., approximately 10 seconds, and an additional maximum tempo difference between the two sequences of factor 3.

6.2. Anchor Note Selection

Throughout our experiments on various refinement strategies, it became obvious that some notes can be annotated with a very high accuracy, i.e., onset time deviations of only a small number of milliseconds, while other ones are much harder to extract and timing errors are therefore larger. This also holds for the onset descriptor based on weighted bandpass filter outputs as described in the last chapter ([Meron and Hirose, 2001] show that repeated notes, for example, are systematically prone to more significant errors). To avoid such larger timing deviations, we propose to determine the timing of notes where the onset is unambiguous. Then, in a second pass, the notes in between and concurrent to those notes are refined while taking this additional information into account.

In general, an anchor note has to satisfy two requirements. On the one hand, its onset time must be extracted with a very high precision, i.e., a timing deviation from its actual onset of only a small number of milliseconds. On the other hand, the extraction must be robust in the sense that the set of selected anchors contains only a very small number of outliers. To achieve these objectives, a refined onset time is obtained for each note using a method which has been shown to be capable of yielding highly accurate results. Then, candidates where confusions or ambiguities are likely due to the tonal context, for example, are dropped, leaving only the most promising candidates.

6.2.1. Candidate Extraction

In [Niedermayer and Widmer, 2010a] as well as [Niedermayer and Widmer, 2010b], the factorization based Pitch Activation feature (see Section 3.4) is used during the anchor note selection. To obtain improved onset time estimates, a search window of length l is centered around the onset time t_{dtw} reported by the DTW algorithm for each note. The parameter l has been chosen to be 2 seconds, since preliminary evaluation of Dynamic Time Warping based alignments has shown that only a marginal number of outliers deviates from the ground truth by more than a second (see Table B.1(a) in Appendix B for details).

Within this search window, a factorization of the spectrogram is performed, taking the tonal context of the note under consideration into account. To this end, a dictionary $\overline{W}_{local}$ is used, consisting of tone models corresponding to only those pitches which are expected to be played within the time span l centered around the current note. It has proven beneficial to use an additional white noise component, in which the energies are spread uniformly over all frequency bins.

The resulting activation patterns H are smoothed using a median filter and used in order to extract onset indicators for each time frame. Such indicators, which are common in the literature and were tested in our system, are:

Activation energy Since activation patterns H are very sparse in nature (even when sparsity is not enforced), activation energies greater than zero are strong indicators for note positions. An onset is reported either (1) at the time where the energy becomes greater than zero or (2) at the time where it reaches its maximum.

Energy slopes The first derivative of the activation energy corresponds to energy changes. Positive slopes, as they occur at note onsets, are filtered by half wave rectification. The extracted onset time is the one where the energy increases the most, i.e., at the maximum of the derivative.

Relative energy slopes Since transients at note onsets are characterized by energy burst across the whole spectrum, other pitches – especially ones with shared harmonics – might show low activation energies during such phases as well. Therefore, the increases in energy of the pitch under consideration in relation to the overall frame energy is also taken into account. The note onset is then set to the time at which the maximum of the relative energy derivative is observed.

Experiments have shown that the maxima of the derivatives are good predictors for note attacks, while the activation energy itself has turned out to be less significant. Comparing the slope of the absolute energy to the one of the relative energy revealed a slight advantage of the relative energy derivative which was, therefore, chosen as onset detection criterion. In contrast to the onset candidate t_{dtw} obtained by the initial alignment, this new estimate t_{nmf} can deviate from other notes with the same score time.

To summarize, the new onset time candidate t_{nmf} is obtained as follows:

1. Compute the tonal context of each note, i.e., select the set I of pitches which are expected to be played within a window of 2 seconds around the note under consideration including an additional noise component.

2. Calculate the Pitch Activations h_i within the search window using only the tone models $\overline{w}_i$ with $i \in I$.

3. Obtain the onset indicators $\hat{\Delta h_i}$ by half-wave rectifying the derivatives of the individual pitches' activation h_i over time.

4. Choose the onset candidate for the pitch $j \in I$ under consideration at the time frame where $\hat{\Delta h_j} / \sum_{i \in I} \hat{\Delta h_i}$ has its maximum.

Considering related work, [Scheirer, 1997] is the one that presents the approach which is most similar to our proposed system. There, onset detection by selective bandpass filtering is described in the context of score supported audio transcription. According to this method, a note is found by summing up the energy in all frequency bands corresponding to the f_0 as well as the harmonics of a pitch. In order to avoid the influence of other pitches with overlapping harmonics, partials that collide with those of an other note struck at the same time are neglected. The note onset is then reported at the time frame where the derivative of this sum has its maximum.

In [Niedermayer, 2009a], a comparison of our system to our own implementation of Scheirer's approach is presented. We use the same computational framework as de-

	rpa	sbf
$25\% < x$	5.6 ms	10.0 ms
$50\% < x$	14 ms	20 ms
$75\% < x$	32 ms	40 ms
$95\% < x$	137 ms	128 ms
$x < 10$ ms	40.0%	24.9%
$x < 50$ ms	85.6%	81.3%

Table 6.1.: Comparison between all anchor candidates for the first movements of the Mozart sonatas computed based on relative Pitch Activation (rpa) and based on selective bandpass filtering (sbf) according to [Scheirer, 1997]

scribed above and only exchanged the factorization based Pitch Activation feature in the refinement step by an onset detector based on selective bandpass filtering. The accumulated results on the first movements of the Mozart sonatas are shown in Table 6.1. It demonstrates that bandpass filtering yields results less accurate than those produced by NMF, and mostly even less accurate than those achieved by the alignment based on chroma vectors. A possible reason is that the STFT based version of selective bandpass filtering relies on just a few frequency bins while NMF takes the whole spectrogram into account.

6.2.2. Candidate Selection

A ratio of notes where the extracted onset deviates from its ground truth timing by less than 10 ms of 40% is promising (see Table 6.1). However, the problem of separating those candidates from the ones where the timing estimation is less accurate remains. Motivated by the assumptions made during the candidate extraction itself, namely that an onset corresponds to the maximum increase in relative Pitch Activation and that the real note onset is close to the estimate obtained by the DTW algorithm, two simple rules can be formulated.

When thinking of repeated notes or of fast passages in which a certain pitch is played several times within the search window, it becomes obvious that selecting the maximum increase of relative Pitch Activation is too simple to yield meaningful results. However, estimating the onsets of repeated notes is a relatively hard problem in itself. Spectral energy of a sustained note weakens the indicators for the onset of a new note if they have the same pitch. Under these circumstances, algorithms are likely to get mislead by onsets of other notes with overlapping harmonics. This fact makes such notes ineligible to be anchor notes, as a high confidence in the exact estimation of the onset time is essential. Thus, all notes which are played twice or even more often within the time

span l of the search window, as determined from the score, are discarded from the anchor candidates.

Likewise, all notes are dropped from the list of anchor candidates, for which the initial onset estimate t_{dtw} and the estimate given by the factorization-based feature t_{nmf} differ by more than a certain time span which could have plausibly been caused by an arpeggio or a simple asynchrony. This is justified because such a conflict decreases the confidence in the onset estimation. Moreover, there is no safe way to give either t_{dtw} or t_{nmf} a preference over the other. On the one hand, t_{dtw} is supposed to be more robust, since much more context information is incorporated. On the other hand, t_{nmf} is not bound by the constraints inherent to the DTW algorithm, and therefore able to yield more accurate results. During the anchor extraction, the semantics of the search radius l is thus reduced to defining the tonal context of a note rather than being an actual search window.

In summary, the two times t_{dtw} and t_{nmf} are calculated by the DTW algorithm and finding the maximum slope within the factorization-based Pitch Activation (see algorithm above). A note is then selected as an anchor if the following two criteria are met:

1. $|t_{dtw} - t_{nmf}| < \text{threshold}$

2. there are no other notes of the same pitch within $t_{dtw} \pm l/2$

In our specific implementation, the maximum difference $|t_{dtw} - t_{nmf}|$ allowed between the two onset estimates was set to 20 frames, i.e., a little more than a tenth of a second at a hop size of 256 samples.

A detailed evaluation of this anchor extraction step is presented in Appendix B. There, for each piece as well as the entire corpus the timing deviations between detected and actual note onsets are given for anchor and non-anchor notes. It is shown that while 92% of all anchor notes are reported within a 50 ms range around the correct onset and 65% of all anchors are accurate at a 10 ms tolerance (see Table B.1(b)), these values are only 66% and 23% respectively for the non-anchor notes (see Table B.1(c)). It is remarkable that the accuracy of anchor notes is approximately the same for each movement, while the DTW-based initial alignment is significantly less accurate for the second movements in comparison to the others (see Table B.2 – Table B.4). Taking into account that 32,552 notes, i.e., approximately 32% of all notes, were selected as anchors, this step would be a major improvement of alignment accuracy on its own.

6.3. Between-Anchor Refinement

After extracting the anchor notes, the remaining notes have to be revised. For each of them (with the exception of notes played before the first or after the last anchor notes) the span of time during which it can be played is clearly constrained by the preceding and the successive anchor.

6.3.1. Beta distribution

In addition to a new search window, bounded by the nearest anchors, rhythmic information in the score can be exploited to make even more detailed predictions on where to look for an onset. Therefore, the numbers or fractions of beats between the anchor notes and the note n under consideration are extracted and their relation is transferred onto the timescale of the audio recording. We assume that the anchor note n_0 (detected at performance time t_0) is the anchor directly preceding the note n and the anchor note n_1 (detected at performance time t_1) is its directly successive anchor. Then the onset of n is expected to occur at $t_n = t_0 + c(t_1 - t_0)$. The relative offset c can be inferred from the score. If, for example, a note is supposed to be played exactly one beat after an anchor note and the next anchor note follows at the interval of another exact beat, then the offset $c = \frac{1}{2}$. If, in another example, a note is notated one beat after an anchor but the next anchor follows after only half a beat, then the offset $c = \frac{2}{3}$. In the same manner, the values $c = \frac{1}{3}$ and $c = \frac{2}{4}$ are obtained for the two examples shown in Figure 6.2.

To account for inexactnesses of the anchor extraction and expressive tempo changes, the "*expectation strength*" of the onset occurring at time t is modeled by a beta distribution[1]. The beta distribution is defined continuously on the interval $[0, 1]$ and has values of zero outside this range. Depending on the values of its parameters α and β, the density function can take several forms, for example, that of a uniform distribution, it can be strictly increasing or decreasing, U-shaped, or – as in our case – it is unimodal ($\alpha > 1$ and $\beta > 1$). Its density function is defined as

$$f(x)_{\alpha,\beta} = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1} \tag{6.1}$$

[1] The beta distribution was chosen for pragmatic reasons (the flexibility of its shape and its restriction to a fixed interval) rather than for precise probability-theoretic reasons.

where B is the beta function

$$B(\alpha, \beta) = 2 \int_0^{\pi/2} \cos^{2\alpha-1}\theta \sin^{2\beta-1}\theta \, \mathrm{d}\theta \tag{6.2}$$

Mode $\hat{x}$ and variance σ^2 of the distribution are therefore given by

$$\hat{x} = \frac{\alpha - 1}{\alpha + \beta - 2} \tag{6.3}$$

$$\sigma^2 = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)} \tag{6.4}$$

In this application, the parameters α and β are set by fixing a mode $\hat{x}$ and a variance σ^2. The former is assumed to be at the onset time we expect according to score and anchor notes. The linear projection of this time to the domain of the beta function (i.e., $[0, 1]$) is exactly the offset factor c as introduced above.

The variance is chosen such that it allows for expressive variations and inexactnesses of the anchor extraction, but prevents notes from being placed at rhythmically unreasonable timings. Experiments have shown that the value $\sigma^2 = \min(c, 1 - c)/\gamma$ with a "sharpness" factor $\gamma = 20$ results in plausible expectation strengths.

Given the values of $\hat{x} = c$ and $\sigma^2 = \min(c, 1 - c)/\gamma$, the parameters α and β can be computed from Equation 6.3 and Equation 6.4 using mathematical software. The resulting equations, however, are not given here due to the large number of terms they contain.

Two such expectation functions are depicted in Figure 6.2. The upper plot shows the onset likelihood for the onset time of the third note, assuming that the first and the fifth note are anchors. The time span between the anchor comprises three beats. Since the note should be played after the first out of these three beat-to-beat intervals, the function is clearly skewed. This is desirable because a musician's freedom of expressive timing is greater when the score calls for longer inter-onset intervals. The second function is the likelihood of the fourth note's onset time given notes number one and six as anchors. The function is now symmetric, since the onset time given by the score is exactly half the time span (two out of four beat-to-beat intervals).

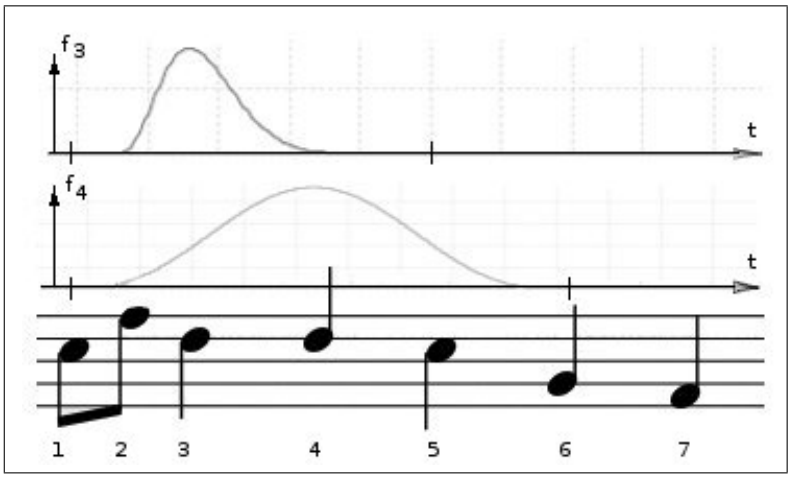


Figure 6.2.: Onset expectation strength for the 3^{rd} and 4^{th} note.

In order to transfer these expectation strength functions from the score into the audio domain, another linear projection is applied. Here, a value $x \in [0, 1]$ is projected to a performance time of $t_x = t_0 + x(t_1 - t_0)$.

6.3.2. Onset estimation

To extract revised onset estimates for non-anchor notes, a standard onset detection algorithm is chosen. The onset indicator used, is extracted from the constant Q spectrogram over the time span in which the onset expectation strength, as described above, is greater than zero. The parameters of the constant Q spectrogram are chosen, such that each energy bin corresponds to a specific pitch. The hop size is set to 256 frames, resulting in a very high overlap ratio at the lower bins. The actual feature extracted from this representation is the energy increase, i.e., the half-wave rectified energy derivative, within the bin k corresponding to the fundamental frequency of the pitch under consideration, i.e.,

$$odf_k(t) = max(0, \hat{X}_k(t) - \hat{X}_k(t-1)) \tag{6.5}$$

where $\hat{X}_k(t)$ is the spectral magnitude of the $\mathrm{k^{th}}$ bin at time t.

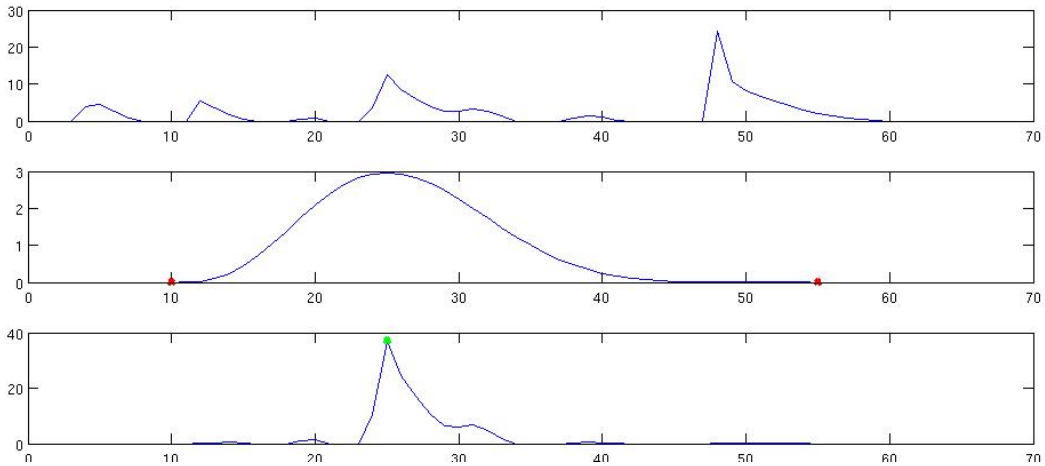


Figure 6.3.: Calculation of the onset indicator function: Compute the half-wave rectified derivative of the bin energy magnitudes (top), place a *beta* window between the two anchors (middle), and obtain the onset estimate (green marker) as the maximum of the product of these two functions. The red markers in the middle plot indicate the positions of the two anchors.

The spectral information is combined with the expectation strength $beta_{\hat{x},\sigma^2}(t)$ by piecewise multiplication. The note onset is then reported at the time frame where this function has its maximum.

$$t_{onset} = \underset{t}{\operatorname{argmax}}\ \mathrm{odf}_k(t)\,\mathrm{beta}_{\hat{x},\sigma^2}(t) \tag{6.6}$$

In doing so, the extraction of note onsets in "difficult" scenarios benefits from additional knowledge obtained from the score as well as the anchor positions. In Figure 6.3 an example for this computation is shown, where the offset c between two anchors at the frames 10 and 55 of this excerpt equals 1/3.

The sole consideration of the fundamental frequency of a note through this refinement step might seem to neglect valuable spectral information. However, the whole spectral information has already been exploited in the anchor extraction where the respective onset candidate was not considered reliable. This can have two reasons – the respective pitch is expected to be played twice within the analysis window according to the score, or the candidate's timing differs from the tentative onset time, obtained by the DTW algorithm, by more than a threshold. Independent from the actual cause of this uncertainty, relying on a simple, nevertheless purposeful feature is a reasonable decision. In piano music, the fundamental frequency is, in general, the most significant peak within the spectrum of a tone. The effect of a *missing fundamental* or a fundamental frequency which is at least damped, as common for instruments where, for example, the corpus suppresses oscillations at very low frequencies, is very unlikely. On the other

hand, given the narrow context the search is focused on, a single partial is less prone to overlapping, and therefore less ambiguous, than a complete series of harmonics.

6.4. Refinement of Notes concurrent to Anchors

A special case are notes which are, according to the score, played concurrently to an anchor. In such a scenario, instead of exploiting an expectation about the timing of a note yielded from interpolation between the two adjacent anchor notes, the onset time of the concurrent anchor can be used as a more reliable cue. Considering two points in time, i.e., events where a match between score and audio representation is known, the alignment between those fixed points is not linearly scaled but subject to changes in local tempo and also intra-chord micro-timings. When, in contrast, a score note concurrent to the one under consideration is already matched to the audio recording, the uncertainty about tempo changes can be avoided. Small differences in the timings of individual chord notes, however, remain an issue, as well as inaccuracies in the alignment of the anchor note.

Independent of intra-chord micro-timings, it is reasonable to assume that the temporal order of notes is not changed such that an event notated before another one in the score occurs after this second event in the performance. Based upon this assumption, the refined onset times of the adjacent notes can be exploited to limit the search for notes known to be played in between. Combining this information with the onset time of the concurrent anchor, an expectation strength function for the onset of the note under consideration is computed. It is defined as a Gaussian window centered around the anchor note with a width equal to the smaller of the two inter-onset intervals between the anchor and its adjacent notes.

Given the onset detection function $odf_k(t)$ for the pitch with its fundamental frequency in the k$^{\text{th}}$ bin as described above, the onset is detected at time

$$t_{onset} = \underset{t}{\operatorname{argmax}}\ \mathrm{odf}_k(t)\, G(t; t_a, \max(t_a - t_0, t_1 - t_a)^2/\gamma^2) \tag{6.7}$$

where t_a is the refined onset of the concurrent anchor note. t_0 and t_1 are the refined onset times of the two adjacent notes. γ influences the variance and, therefore, the "sharpness" of the function's peak. In preliminary experiments a value of $\gamma = 20.0$ was shown to yield good results. An example is given in Figure 6.4, where the anchor concurrent to the note under consideration is detected at time frame 35 of this excerpt

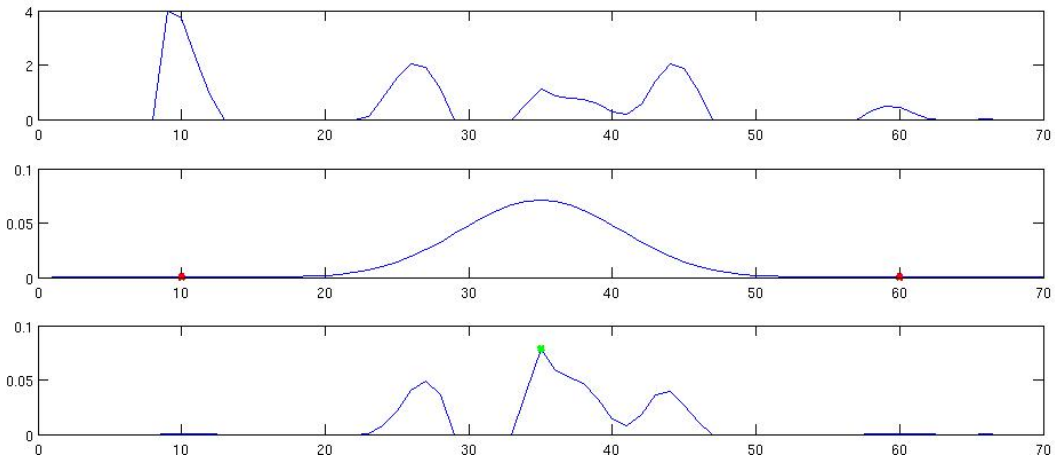


Figure 6.4.: Calculation of the onset indicator function: Compute the half-wave rectified derivative of the bin energy magnitudes (top), center a Gaussian window around the concurrent anchor (middle), and obtain the onset estimate (green marker) as the maximum of the product of these two functions. The red markers in the middle plot indicate the range of the Gaussian window.

and causes the onset to be reported at the smaller peak which also is at frame 35 in favor of the higher peaks at the frames 26 and 44 respectively.

An evaluation of these last two refinement steps can also be seen in Appendix B. The number of non-anchor notes aligned within a 10 ms tolerance range around the actual onsets was increased by a factor of 2 from 23% to 47%. Allowing a time displacement of 50 ms, the amount of correctly aligned notes also increased significantly from 66% to 82% (see Table B.1(c) and Table B.1(d)).

6.5. Evaluation Results

The results of a detailed evaluation as described in Chapter 2 are shown in Appendix B. There, the average, minimum, and maximum of the temporal displacements and their absolute values are given as well as the quartiles, the 5^{th} and 95^{th} percentiles and the number of notes which are accurate when allowing for a 50 ms and a 10 ms tolerance range, respectively. These measures are not only given for the final evaluation result on each individual piece but also for the intermediary results of the initial DTW-based alignment, the selected anchor notes and the non-anchor notes before as well as after their refinement.

Table 6.2 shows the final result over the entire evaluation corpus. Due to our refinement methods, more than half of the notes are aligned with a temporal deviation of less than

10 ms, i.e., an error small enough that it is reasonable to assume that a human annotator would accept the respective note onsets as correct.

avg-error	1.5	(50.5)
std-dev	169.9	(162.2)
min-error	-3460.5	(0.0)
max-error	3776.3	(3776.3)

p-5	**p-25**	**p-50**	**p-75**	**p-95**
-108.1	-8.8	-0.9	10.5	84.4
(0.8)	(4.1)	(9.3)	(24.1)	(232.4)

error<50ms	85.15%
error<10ms	52.41%

Table 6.2.: Alignment result of the proposed system on the entire evaluation corpus where errors are measure in milliseconds. The parenthesized numbers are calculated from absolute timing errors, and the values in the 6^{th} and the 7^{th} row represent the 5^{th} , 25^{th} , 50^{th} , 75^{th} , and 95^{th} percentiles.

Considering the 95^{th} percentile of absolute errors of 232 ms, the number of outliers is small. Only 0.68% of all notes are aligned with a timing deviation of more than one second. Most of these errors occur at the endings of a piece or the endings of a major phrase followed by a pause. These scenarios are characterized by ritardandi and high degrees of polyphony. This combination makes an accurate alignment exceptionally difficult. Due to the reduced tempo there are fewer cues in terms of note onsets the system can use, and a high degree of polyphony results in increased ambiguity when relating partials to notes.

6.6. Conclusion

We have presented a system which is able to align more than 50% of all notes within our evaluation corpus with a temporal deviation from the actual onset time of less than 10 milliseconds. According to [Friberg and Sundberg, 1992] this is about the just-noticeable error humans are able to recognize within series for pulses played at a fixed inter-onset interval. We, therefore, assume alignments of notes with an onset time deviation below this threshold as ”good enough”.

A question suggesting itself is, if a system yielding such a result is useful. On the one hand, one might argue that, especially in the domain of performance analysis, the described errors are too severe. Phenomena, such as intra-chord micro-timings or the

timing of ornamentation notes, where note timings deviate at the magnitude order of a few milliseconds require a level of accuracy which is beyond the capability of our system. Also, when playing back a synthesized version of the alignment result, error become evident frequently. Finally, we have to expect that the system performs worse on more complex signals compared to the Mozart sonatas, such as recordings of pieces with higher degrees of polyphony or increased pedal usage.

On the other hand, except from short segments of up to about 4 seconds where the alignment system fails locally, the extracted note onsets are within a range around the actual onset which is small enough to allow for an unambiguous correlation and a fast correction. This is consonant with our objective of minimizing the effort of annotating musical performances. In the next section a tool will be described which allows for the inspection and manual correction of automatically computed alignments. Also, for a number of applications where not the exact timing for each note is required (such as certain visualizations) the annotation effort can be further decreased.

In summary, we consider our results promising, taking into account that in the presence of playing errors or intended deviations of the performance from the score not even symbolic alignment is an easy task.

7. Applications

7.1. Graphical Annotation Tool

The system for audio alignment, as proposed in this thesis, aims at a high accuracy in terms of time displacement of the note onsets. Although results are promising, the requirements of some applications in the context of computational musicology cannot be met. A consistent temporal accuracy at a few-millisecond level is a prerequisite for, e.g., an in depth analysis of expressive timing. With regard to the fact that current state-of-the-art alignment systems do generally not yield such precise results, one of the objectives while developing our system was to minimize manual post-processing efforts. This is reflected in the evaluation metrics, where tolerance ranges for time displacements and percentiles were preferred over average timing errors.

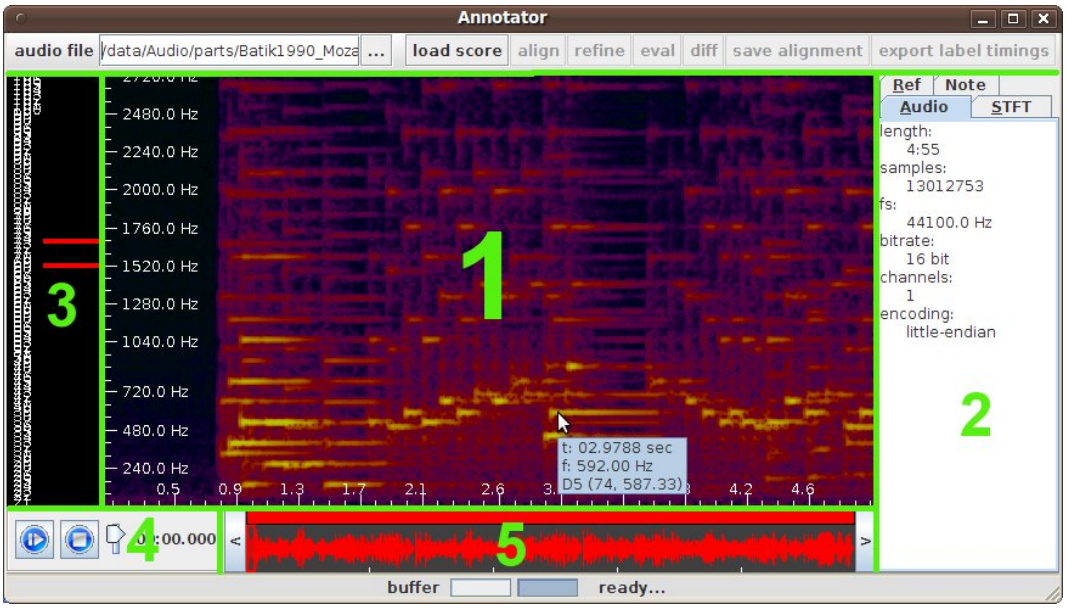


Figure 7.1.: Screen-shot of the annotation tool showing (1) the spectrogram of a loaded audio file, (2) meta information about the audio file format, parameters of the applied STFT, and individual notes, (3) a model-based decomposition of the spectrogram at the current cursor position, (4) player controls, and (5) a navigation panel indicating the current position within the entire audio recording.

Figure 7.1 shows a screen-shot of the user interface of the annotation tool which was built for the inspection and correction of audio-to-score alignments. As soon as an audio file is loaded, the user can acquaint herself with the piece by playing it back (4) or browsing through the spectrogram (1,5). While doing so, the factorization panel (3) shows a dictionary-based decomposition of the audio frame at the cursor position. Red bars indicate the activation strength of individual pitches at the given time. In Figure 7.1, one can see the beginning of a performance of Mozart's piano sonata k.279 with a cursor position between the individual note onsets of the d-minor chord at the first beat of the second measure. The notes already sounding – the A4 (pitch number 69) and the D5 (pitch number 74) – are correctly detected and indicated in the factorization panel.

After loading a score in MIDI format, it can be automatically aligned to the audio recording by either a fast or a more accurate algorithm. The first one computes Chroma features from the audio as well as from the score representation and performs Dynamic Time Warping at a relatively low temporal resolution. The more accurate method is based upon this initial alignment, but, in contrast, also includes the multi-pass refinement mechanisms as proposed in Chapter 6. Alternatively the user can also load an arbitrary alignment computed during an earlier session or by another software.

The main working area throughout the inspection or correction process is the panel showing the spectrogram of a loaded audio file overlaid with the aligned notes. By moving the cursor over a note, its parameters are shown in the meta-data area, the activation energy of the respective pitch over time is plotted overlaying the spectrogram, and a ruler indicates the frequencies of the ideal harmonics (see Figure 7.2). Using this information, the user can correct the alignment by moving, stretching, or compressing note objects in time by simple dragging the boundaries of the respective graphical note objects. For the verification of an annotation the audio recording can be played back in conjunction with an automatically overlaid click-track indicating the current note onset positions. The result of this post-processing step can be stored to an arbitrary file.

Using this tool we hope to be able to considerably speed up the annotation process of audio recordings at a note level. However, even in the scenario where most notes are aligned with sufficient accuracy, the verification of the exact note onset timings is still a laborious task. As a compromise between the amount of data acquired and the effort required to do so, meta events can be inserted into the MIDI file directly after a Note-On event. Such meta events having the same time stamp as a note will not be separated from this note during the alignment process. Thus, one can label note events of interest and export the corrected timings of the corresponding labels only. This is a

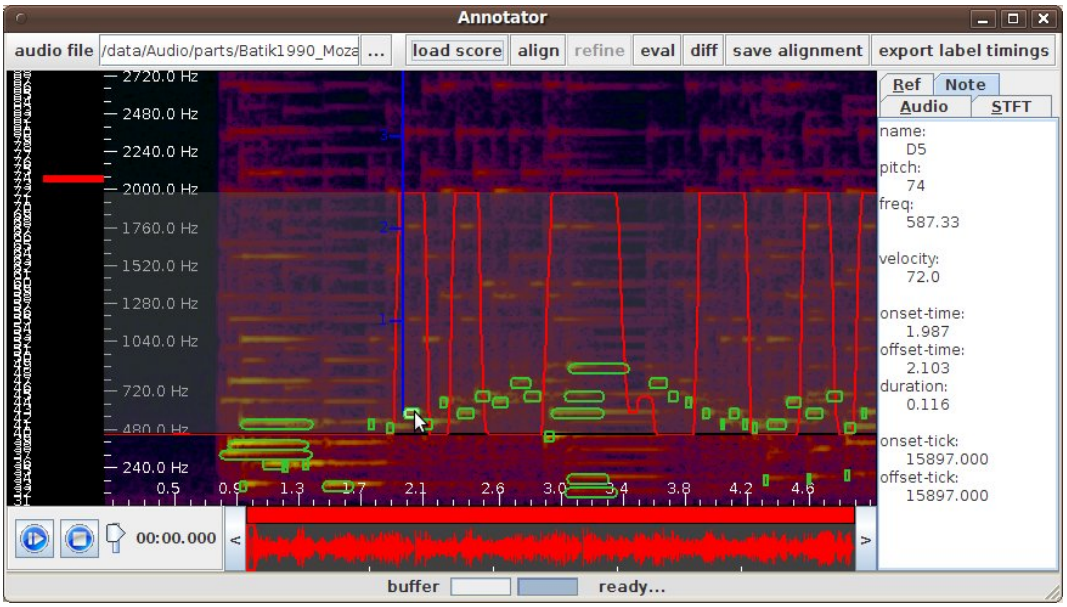


Figure 7.2.: Screen-shot of the annotator tool, showing an aligned score and the activation energy of the highlighted note over time (red function) as well as its ideal harmonic frequencies (blue ruler)

useful feature when one is, for example, interested in the voice carrying the melody or in an annotation at a bar or beat level.

7.2. Musical Performance Research

Assuming an accurate annotation of a performance of a piece, one is able to conduct an in-depth analysis of the expressive variations an artist makes during the performance. While, at the current state-of-the-art of music signal processing and content based music information retrieval, nuances at a level of intra-chord dynamic variations or articulation cannot be distinguished automatically, the extraction of tempo curves at a beat level can be achieved – at least for the Mozart sonatas – with an affordable amount of manual inspection and correction. On the other hand, visualization approaches are important applications of the methods described throughout this thesis. There, a note onset accuracy down to a few milliseconds is often lost due to data reduction, smoothing, or summarization. This further reduces the need for manual corrections.

7.2.1. Performance Visualization

Tempo Curves

From the alignment of two audio recordings of performances of a specific piece of music one can only infer the tempo relation for each segment of the piece. However, without additional knowledge about the score of the piece or, to be more specific, the inter-onset interval of any two notes in terms of score time, i.e., beats, a measurement of absolute tempo cannot be made. In the context of Audio-to-Score Alignment, a symbolic representation of the score is inherently available. Nevertheless, this does not guarantee that the above requirement is met.

The issue of richness and quality of the score representation used for an Audio-to-Score Alignment has not been discussed yet. Although the algorithm benefits from a score representation comprising the (relative) duration between two events, for an alignment using Dynamic Time Warping it is, in principle, sufficient that the correct temporal order of note events is known. Other methods, such as Hidden Markov model based approaches, explicitly exploit knowledge about the expected temporal interval between two events. Another example of a system which requires the score to contain information at this level of detail is the multi-pass refinement process proposed here. Nevertheless, none of these algorithms use concepts such as note values and beats or time signatures and bars.

However, for the measurement of the absolute tempo in units of beats-per-minute (BPM), those concepts are crucial and a closer inspection of the used score representation is required. The MIDI file format specifies time and note values in terms of ticks, i.e., events from a virtual timer, and a beat subdivision rate, i.e., the number of such ticks per beat. The tempo of a playback of this file is then determined by the period of the timer, which can be set and changed using MIDI meta events.

While this information seems sufficient to extract the required information, problems arise when the MIDI file also encodes tempo changes during a piece. This can be due to performance cues provided by the composer of a piece or global tempo indicators which are often given at beginnings of main segments, such as whole movements. While a good practice is to encode those changes by meta events and changing the period of the timer, in a large number of MIDI files one can find a global tempo and arbitrarily modified event timings. In such cases, it is not possible to infer beat information from the MIDI ticks. In the Mozart corpus, however, we do not only have this information but also the detailed alignment to a score including further annotations.

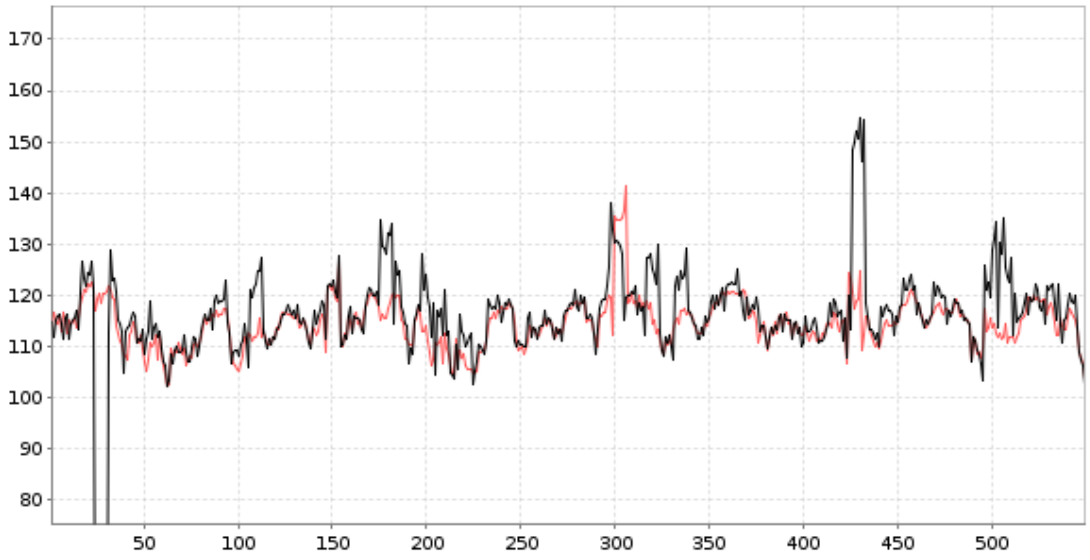


Figure 7.3.: Comparison between actual performed (red) and automatically extracted (black) tempo

To estimate the tempo, those notes are chords are considered which are played exactly on a beat. For beats where no note is played, a beat time is approximated by interpolation. The tempo is calculated as $60/IBI$, where IBI is the inter-beat interval. For beats on which only one note is played, this is simply the time difference. If a chord is played on one of the beats and a single note on the other, the chord note with the pitch best matching the pitch of the single note is selected for the IBI-calculation. This is due to the assumption that those two notes belong to the same voice and that a better tempo estimate can be obtained by observing one voice only. If chords are played on both beats which are compared, each note of the chord with the lesser degree of polyphony is matched to a note of the other chord. The final inter-beat interval is obtained by taking the average over individual values the corresponding to such pairs of notes. The remaining unmatched notes of the chord with the higher degree of polyphony do not contribute to the tempo estimation.

Figure 7.3 shows the tempo curve automatically extracted from the Audio-to-Score Alignment of the piece k.279-1 in comparison to the actual tempo computed from the ground truth data, both smoothed over a window of 7 beats. One can recognize two passages with significant errors – one around the 25^{th} beat where the detected tempo decreases down to 25BPM and one around the 430^{th} beat where the actual tempo is overestimated. The alignment errors responsible for these misestimations of the tempo can be easily detected in Figure B.1. For the rest of the piece the errors are less severe and meaningful tempo estimates are obtained for long segments.

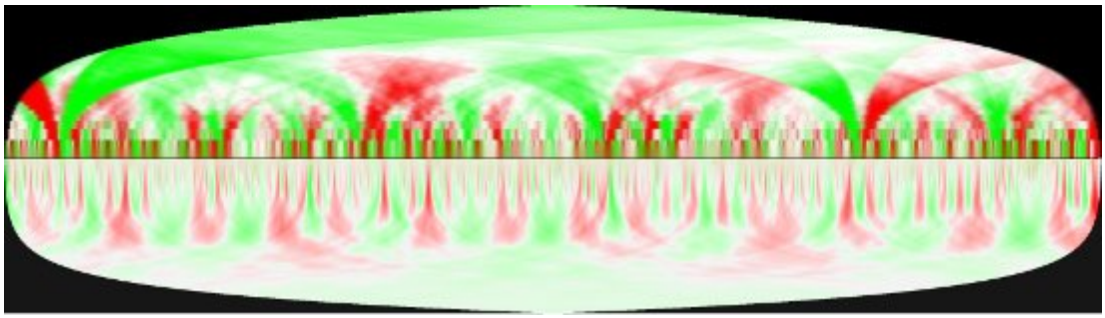
Tempograms and Dynagrams

While the tempo curve is an adequate means to analyze a single performance, it is not very intuitive when comparing two or more performances. [Langner et al., 2000] describe *Dynagrams* as a visualization of dynamic trends over increasingly long periods of time. There, the performance time and the period of time over which a trend is observed are displayed along the axes of a two-dimensional space. Crescendi are visualized by a corresponding saturation of green, while decrescendi are assigned the color red. If dynamic changes cannot be observed at all or only to an insignificant extent, the saturation of green or red is reduced to zero, resulting in white areas. If, at a certain playing time, a dynamic change is not only a local phenomenon, but part of a more general trend throughout the performance of a piece, arch-shaped structures can be observed in the dynagram as the span of time under consideration increases.

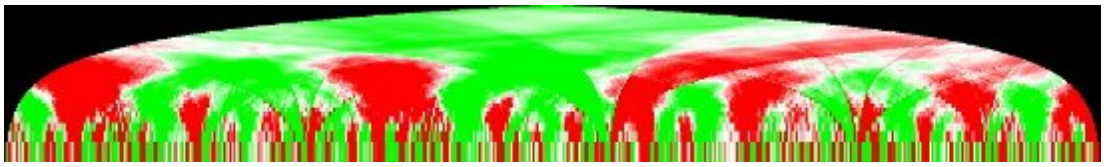
The computation of the dynamic change over a certain period of time is based on the loudness curve of the performance. Such curves can easily be calculated form the audio signal using arbitrary models of perceived loudness. To allow for analysis at different levels, smoothing is applied using mean-filters at respective length. The gradient of the filtered curves are then visualized using the color map as described above.

The resulting visualization provides an overview of the performance in terms of dynamic phrasing at various levels. On the one hand, the intensity of such expressive variations can be directly seen from the colors' saturation. On the other hand, differences between two interpretations concerning the organization of a piece into phrases at different levels become obvious from the arch shaped structures emerging in the dynagram.

Assuming that a tempo curve can be obtained from arbitrary performances of a piece, the idea of the dynagram visualization can be transferred to the tempo concepts accelerando and ritardando. As described above, this can be achieved by Audio-to-Score Alignment, given that the note timings in terms of beats can be deduced from the score representation. Alignment errors, i.e., timing deviations, can be tolerated up to a certain extent, bearing in mind that for the visualization of general trends, the tempo is smoothed over a span of time of several seconds. Therefore, for the generation of tempograms (analogous to the name *dynagram*), only those outliers with a significant temporal deviation from the actual onset need to be corrected manually. As shown in Figure 7.3, there are mainly two significantly erroneous regions in the alignment of the piece k.279-1 – one around the 25^{th} and the other around the 430^{th} beat. The automatically extracted tempogram as shown in the upper half of Figure 7.4(a) shows incorrect trends at these passages, whereas the other trends can be recognized correctly compared to the tempogram computed from the ground truth data as shown in Figure 7.4(b).



(a) Combined Tempogram (upper half) and Dynagram (lower half)



(b) Tempogram computed from ground truth data

Figure 7.4.: Combined Tempo- and Dynagram representation as extracted automatically (a) and the Tempogram computed from the ground truth data (b)

Another visualization which combines tempo and loudness curves would be the *performance worm* (see [Dixon et al., 2002] or [Langner and Goebl, 2003], for example). There, during the playback of a performance, the current points in the tempo-loudness space are connected over time. Having the older values fade out, results in a 'worm' which moves according to the two performance parameters. Since tempo and loudness of a performance can change abruptly, extensive smoothing is needed to generate intuitive worm movements. This favors the usage of data obtained by automatic alignment of the performance to its respective score, since errors, as for the tempogram visualization, can be tolerated up to a certain extent.

7.2.2. Expressive Performance Rendering

Visualizations are an adequate method to explore a data corpus and to perform a qualitative evaluation of expressive variations. A more sophisticated issue is the automatic extraction of measurable performance characteristics in terms of generalizable rules. One approach to study musical performance as a creative process is to build systems which are able to automatically render an expressive performance of a piece.

To encourage work in this field, an initiative named *Rencon*[1] (Musical Performance Rendering Contest for Computer Systems) was founded that organizes a series of workshops and competitions between performance rendering systems. There, computer systems

[1] http://http://renconmusic.org/

are required to read in the score of a yet unseen piece of piano music and produce an expressive performance of that piece in MIDI format. The participants themselves are given one hour of processing time, in which they are only allowed to ensure that the score was parsed correctly and to add their own annotations to the score. They are not permitted to listen to the result or to do any post-processing, especially not at the level of output MIDI notes. One exception from this strict policy is the opportunity to look at the key movements while the silenced piano is playing back the resulting performance. This is necessary to give participants the chance to correct minor errors in cases where the automatic system has failed for some reason.

The system which won all prizes in the latest two RENCON contests[2] is called *YQX* and relies on a Bayesian performance model (see [Widmer et al., 2009] and [Flossmann et al., 2011]) to alter certain performance parameters locally, i.e., on top of a general trend which is determined by specifications provided within the score. The basic idea is to use statistical machine learning methods to establish certain relations between features which can easily be obtained from the score and the most influential performance parameters. After breaking this approach down to the note level, the task is to observe the score context of each note and determine three target variables y_i – the (1) timing, in terms of a ratio of the inter-onset interval as given in the score in comparison to the respective time interval in the performance; (2) dynamics, i.e., the relative loudness of each individual note; and (3) articulation, measured as the relation between how long a note is sustained and the corresponding prescription by the score.

The features computed from the score are separated into two groups. On the one hand, there is a set Q of discrete features, such as the pitch interval to the following note or an abstract description of the rhythmic context, measuring the durations of the note under consideration and the two adjacent ones in terms of *long* and *short*. On the other hand, properties, such as the duration ratio of a respective note compared to its successor or the musical closure as obtained by implication-realization analysis, form a set X of continuous score features.

The Bayes net, as outlined in Figure 7.5, models the target variables as dependent on the combination of those discrete and continuous features $p(y_i|Q, X)$, where the evidence Q and X have marginal likelihoods $P(q_j)$ determined from probability tables and $p(x_j)$ following Gaussian distributions. The model is trained starting with learning distributions $p(y_i|X)$. Then, the dependency on the set of discrete variables Q is estimated for each possible combination of values separately.

[2] held as workshops in the context of the International Conference on Music Perception and Cognition (ICMPC) 2008 in Sapporo, Japan, and the Sound and Music Computing Conference (SMC) 2011 in Padova, Italy

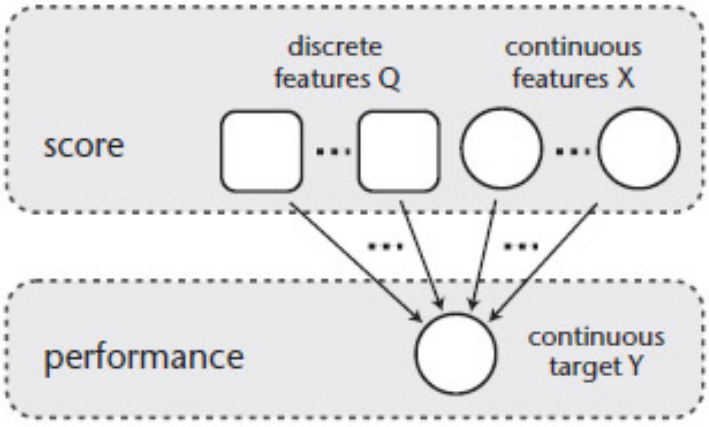


Figure 7.5.: Overview of the structure of the Bayesian model used in the YQX system [Widmer et al., 2009]

As explained in [Widmer et al., 2009], this learning procedure requires an extensive amount of training data. In the example of *YQX* two data corpora have been available, which were sufficiently large to yield meaningful predictions. One corpus – the Mozart sonatas – is the same as used for the evaluation of algorithms performed within the scope of this thesis. The second one consists of 39 selected pieces by Chopin (comprising about 24,000 melody notes) performed by Nikita Magaloff on a computer-monitored grand piano, where the authors have semi-automatically linked each performed note to its respective appearance within the score (cf. [Flossmann et al., 2010]). Here, Audio-to-Score Alignment can be a means of making performances by other pianists available for model training. The automatic performance rendering system would not only be able to imitate performances of works by other composers and their era-specific characteristics, but also to compare different pianists by means of the respective generalized models. Insights into the phenomenon of expression arising from this field of research (see [Flossmann and Widmer, 2011] or [Grachten and Widmer, 2011], for example) could then be based upon a broader data base.

7.3. Audio-to-Audio Alignment and Structural Analysis

While Hidden Markov model based alignment approaches, as described in Chapter 4, are inherently based upon a symbolic score representation, others, such as Dynamic Time Warping based systems, do not require high-level descriptors. Particularly, rendering the score representation using a software synthesizer was introduced as a method to obtain the same audio features (Chroma vectors, for example) from the score as from the recording of a performance. Doing so reduces the problem of Audio-to-Score Align-

ment to Audio-to-Audio Alignment, where both instances of a piece of music are given in a uniform representation.

However, in addition to aligning a musical recording to a corresponding score, Audio-to-Audio Alignment can be applied to a number of dedicated tasks. These do not only include Audio Retrieval tasks such as (Cover) Version Detection, but also Song Structure Analysis. There, instead of comparing two feature sequences, individual frames of one musical representation are compared to each other. In the following, we describe a system for the automatic detection of repeated sections within a song that was developed to demonstrate the use of alignment techniques for structural analysis.

Figure 7.6 shows a music player that we have implemented integrating a song structure analysis layer, allowing the user to navigate directly to a certain repeated section. To this end, the user is provided with a graphical representation of such repeated passages. The horizontal axis of the navigation panel (on the lower right) represents the playing time in terms of frames, whereas the rows correspond to certain musical themes which are played during the highlighted segments.

The algorithm used to derive this information is strongly based upon the method proposed by [Goto, 2006] with some minor simplifications. In analogy to a large number of Audio-to-Score Alignment systems, a (dis-)similarity matrix is computed using Chroma vectors and the (inverse) Euclidean distance as described in Section 3.2 and Section 4.1.1. However, instead of considering two different feature sequences, the audio recording is compared to itself.

The detection of repeated sections can then be reduced to finding line segments within the similarity matrix along which high values can be observed. In addition, these line segments have to be parallel to the main diagonal. This constrains repeated sections to be played at the same tempo. According to [Goto, 2006], such sections are found efficiently by first dropping the redundant values within the similarity matrix, reducing it to a triangular matrix. After doing so, it is converted, such that the vertical axis no longer represents the playing time, but the time lag between two audio frames. The value of a point (t_0, t_1) is written to $(t_0, t_0 - t_1)$. This transformation into the time-lag space has the advantage that repeated sections, where corresponding frames share a uniform time-lag, are represented as horizontal lines. This self-similarity matrix is also visualized in the graphical user interface of the music player shown in Figure 7.6.

After applying a local threshold, suppressing noise and emphasizing line segments with high similarity, the average over each row of the remaining matrix is computed. A peak in the average similarity function over all rows, i.e., time lags, indicates a probable lag between two repetitions of a same theme. However, before examining the corresponding

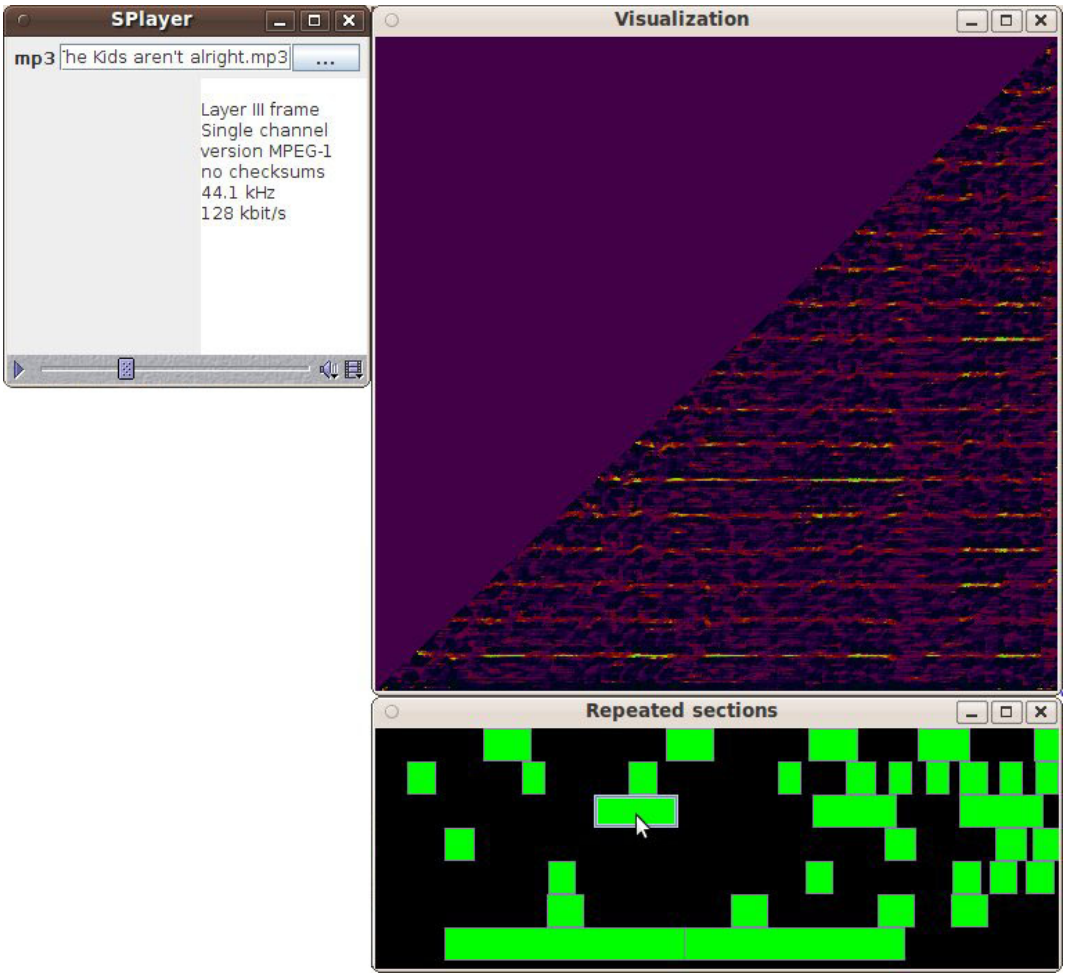


Figure 7.6.: Screen-shot of a music player with integrated piece structure analysis

matrix rows is detail, a peak picking step is performed. High peaks are separated from low peaks by maximizing the discriminant criterion measure, defined as

$$\sigma^2 = w_1 w_2 (\mu_1 - \mu_2)^2 \tag{7.1}$$

where w_1 and w_2 are the relative class occurrences and μ_1 and μ_2 represent the means of similarity values in each class.

Within the remaining peaks, i.e., matrix rows corresponding to self-similarity values at a certain time lag, the actual repetitions are then located by smoothing and thresholding. In addition to these search and filtering steps, segments with a length shorter than a reasonable minimum value (typically a few seconds) are also discarded. In a final post-processing step, numerous repetitions of a single segment are summarized into one set.

In a qualitative evaluation of this simple algorithm we found that although a considerable number of repeated sections was correctly detected, most of the time at least one of a specific theme's repetitions was missed. Nevertheless, we were able to show that using the techniques described within this thesis (especially Section 3.2 and Section 4.1.1) a basic structural analysis can be performed. Other approaches in this direction, which also partly rely on techniques discussed here, can be found in , [Dannenberg and Hu, 2002], [Lu et al., 2004], [Ong et al., 2006], [Peeters et al., 2002], or [Shiu et al., 2006] for example.

7.4. Version Detection

Version Detection is the task to identify different performances or different interpretations of one piece of music within an arbitrary set of audio recordings. Its main application is the identification of cover versions or plagiarisms. However, it can also be used to recognize the piece a performer plays by relating it to synthesized versions of (the scores of) potential candidate pieces. In such a scenario, most state-of-the-art systems perform some variant of audio alignment as described in Chapter 3 and Chapter 4 of this thesis. The reason is that an alignment between possibly matching sections within two recordings is a prerequisite for the computation of their similarity.

A detailed overview of current version detection systems is given in [Serrà et al., 2010]. Also, an annual comparison of different algorithms is carried out as part of the MIREX[3] contest. The evaluation is performed on two test sets, one comprising pieces of popular music, the other one consisting of performances of Chopin's mazurkas by different pianists. Best results were obtained in 2009 by [Serrà et al., 2009] and [Ravuri and Ellis, 2010]. Both approaches are based on Chroma descriptors. [Serrà et al., 2009] used cross recurrence plots based on a state space representation of two songs to rank songs according to their similarity. [Ravuri and Ellis, 2010] calculated three different similarity features – two based on cross-correlation and one based on Dynamic Programming – in combination with three different tempo assumptions and trained a Support Vector Machine on this data.

As part of the work presented in this thesis, the described alignment tools were adapted to perform Audio-to-Audio Alignment in a concrete use-case. In cooperation with the Department of Musicology[4] at the University of Vienna[5], the feasibility of version detection for pieces performed by historical musical automata (e.g., musi-

[3] MIREX: Music Information Retrieval Evaluation eXchange (organized by the IMIRSEL at the University of Illinois at Urbana-Champaign) `http://www.music-ir.org/mirex`

[4] http://musikwissenschaft.univie.ac.at/home/ (in German)

[5] http://www.univie.ac.at/en/

cal boxes, 'flute clocks', violin playing automata, barrel organs,...) was studied (see [Niedermayer et al., 2011b]).

The need for such an application arose from the effort to catalog a large collection of recordings of a variety of such automata which was compiled by the *Phonogram Archive* of the Austrian Academy of Sciences [6] over the past 30 years[7]. The thousands of collected recordings represent a large repertoire of music which was popular during the 18^{th} and 19^{th} century. The respective genres range from opera arias to folk songs. Musical automate were a common means of playing back music before the invention of the phonograph. The collection at hand is thus a valuable data corpus for the study of musical trends, such as common tastes or popular repertoire, during those periods of the pre-phonograph era.

One major difficulty with the collection of audio recordings is that, in contrast to the instruments themselves, the performed pieces are documented insufficiently and are often unknown. Nevertheless, it can be assumed that there exist multiple versions and various performances of individual pieces. Therefore, the use-case for an automatic version detection is to cluster the recordings such that different interpretations of one piece are grouped together. After the identification of associated audio recordings, they can be described by a unified set of meta-data.

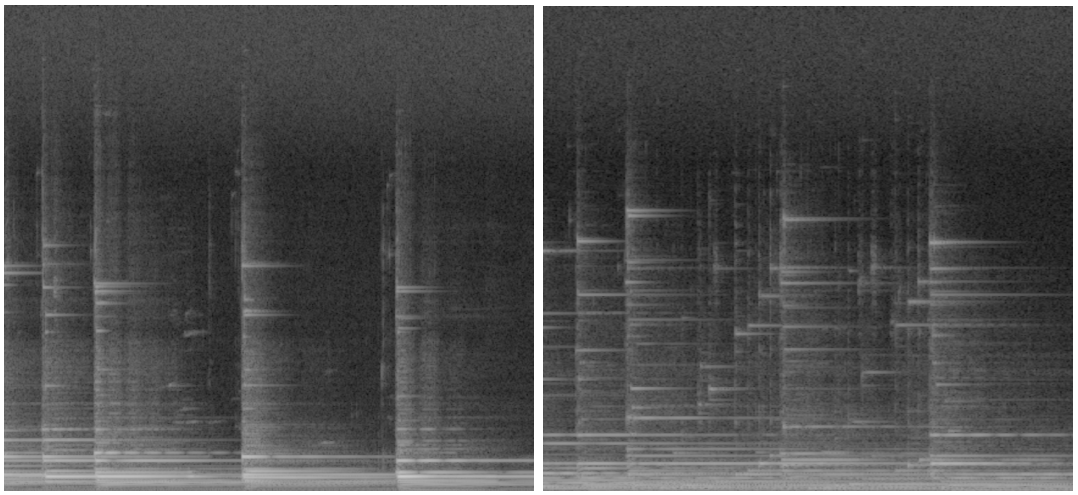
In the following, a pilot study is presented that starts with a small collection of recordings of musical boxes and flute clocks, and with 3 pairs of recordings which are known to pertain to the same composition. Relying on Chroma features and Dynamic Time Warping, by using the Off-Diagonal plausibility measure (see Section 5.2.3) a reference system based on the relative path cost is outperformed.

7.4.1. Acoustic Characteristics of Musical Automata

Musical automata come in wide varieties. Computer controlled pianos and similar modern instruments aside, there are musical boxes, flute clocks, violin playing automata of various kinds, barrel organs, etc. – each of them having individual characteristics. While flute clocks, for example, are largely consistent with the spectral envelopes one would expect from wind instruments – i.e., harmonics at the odd integer multiples of the fundamental frequency – musical boxes, where metal plates are struck, are highly inharmonic.

[6] http://www.phonogrammarchiv.at

[7] A report on the background of this project can be found at http://www.phonogrammarchiv.at/Mechanical_ Music/mechreal.html.



(a) (b)

Figure 7.7.: The ending of the same theme from Mendelssohn's oratorio *Elias* played by two different musical boxes.

The second issue are different arrangements of a same theme for different automata. Besides transpositions into different keys, additional ornamentations or arpeggiations can alter the sound impression of a piece significantly. Figure 7.4.1 shows spectrograms of the ending of a theme from Mendelssohn's oratorio *Elias* as played by two different musical boxes. While in (a) only the main melody notes are played, (b) features luscious ornamentations that make it hard even for human listeners to hear the relation to the main theme.

A third challenge for a version detection system are major changes in the structure of the piece. As described in more detail in the discussion of the evaluation results (see Table 7.1), when two recordings relate to the same piece, this does not necessarily mean that they both comprise the same musical sections. In the data at hand there are samples where one musical box plays only the second half of what another one is playing. Moreover, some recordings are *potpourris* of several themes, such as popular melodies from an opera. When this is the case, corresponding pairs of recordings will only match in part.

7.4.2. Version Detection System

We propose a system working in two steps. First, features are extracted from the audio signals of the individual recordings. Then, a similarity measure is obtained, such that pairs of audio files can be ranked accordingly. To account for transpositions and major structural changes, each piece is split into several chunks of a fixed length

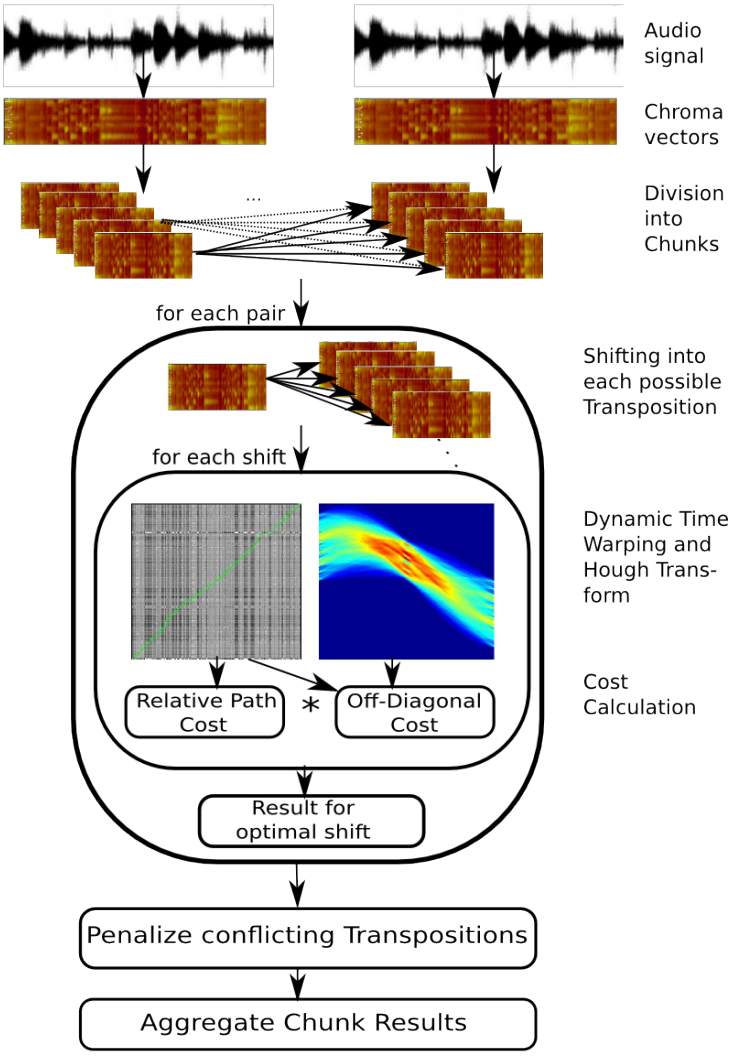


Figure 7.8.: Calculation of the similarity measure

which are then aligned to each other considering all possible transpositions. The final similarity measure is obtained by accumulating the fitness ratings of these alignments. An overview of the whole process is given in Figure 7.8.

7.4.3. Feature Extraction

Since Version Detection generally includes an Audio-to-Audio Alignment step, the features described in Chapter 3 can be applied in this context as well. However, in this scenario, transcription- or onset detection-based features are less preferable. Their computation involves fragile decision processes, which is not justified since two audio recordings are compared and high temporal accuracy is not a main issue. Based on this consideration, 12-dimensional Chroma vectors were chosen as feature representation.

This is concordant with a comparative study presented in [Hu et al., 2003], showing that Chroma vectors outperform several other features, such as MFCCs or pitch histograms, in an audio matching task.

In our specific implementation, we found an STFT with a window length of 4096 samples and a hop size of 1024 samples to be a good trade-off between a frequency resolution fine enough to also resolve relatively low pitches and a time resolution enabling an accurate alignment between two pieces. During the mapping between the STFT's frequency bins and the pitch classes, a cosine window with a width of 1.5 semitones was used to account for the proximity between a bin's center frequency and the fundamental frequency of a certain pitch.

A common deviation of a version of a piece from its original is a change of key. Often, the reason for these transpositions is to adapt a piece to a different instrument. Transpositions can also be motivated by artistic considerations, such as to give a piece a different mood.

Most similarity measures will inevitably fail if the main key has been changed between two versions of a piece. The one used here, for example, depends on the Cosine distance $d_c(A_i, B_j)$ (see Equation 4.2 in section 4.1.1) that measures the error made when matching two Chroma vectors A_i and B_j. The Cosine distance, however, is not robust to transpositions, i.e., shifts of one of the Chroma vectors.

To compensate for possible changes of key, when computing the similarity measures between two feature sequences, each of the 12 shifts is considered. The one yielding the best result is kept and its deviation from the original key is remembered.

7.4.4. Segmentation

Listening to many recordings of different musical automata has revealed that the lengths of the themes played by these instruments vary significantly from less than 30 seconds up to several minutes. This can also be the case when the original piece or even the particular theme was the same. For instance, there are pairs of musical boxes where one performs only the second half of what the other one is playing.

To address this problem, the recordings were split into fragments of equal length. Instead of comparing two whole pieces, each combination of two such chunks was considered. This might seem inefficient from a computational costs point of view. However, the effects are moderate since the effort required to run the alignment algorithm (which is of complexity $O(n^2)$) and compute the similarity measure (including operations of

complexity $O(n^3)$) is reduced accordingly. In addition, the processing of pairs of fragments is fully parallelizable and also, due to the fixed chunk size, the amount of memory needed is limited and independent of the actual length of the full audio recordings.

A fragment length of 25 seconds and an overlap ratio of about 50% have been found to yield good results. The overlap ratio was adapted slightly, such that the last fragments are positioned at the very end of each piece and there is no remainder left. When trying to align pairs of fragments from two pieces, all possible transpositions are considered, and different fragments from the same piece are allowed to be transposed by different numbers of semitones. Conflicts arising from this are handled later when the results from the individual fragment pairs are merged.

7.4.5. Alignment and Similarity Measurement

To calculate a similarity measure, two sequences of features have to be aligned first. To do so, the Dynamic Time Warping algorithm, as described in Section 4.1, is applied. Given such an alignment, an intuitive similarity measure for two pieces of music would be the average cost along the alignment path. However, preliminary experiments have shown that this measure is too simple. One the one hand, it allows for insertions or deletions of notes and thus a change of melody up to certain degree, while on the other hand, it penalizes a change in the structure of a piece. In addition, although Chroma vectors are relatively robust to these effects, higher average alignment costs can still be caused by changes in instrumentation or accompaniment.

In Section 5.2.3, a number of alternative methods to measure the plausibility of an alignment between two feature sequences were described. A relatively simple one is the *Relative Path Cost*, where deviations as mentioned above are partly absorbed by considering the average alignment cost over the entire dissimilarity matrix as an estimator for the average path cost of a random alignment. The quality of an obtained alignment is then compared to, i.e., divided by, this baseline.

Instead of analyzing the cost of an alignment, we have proposed (cf. Section 5.2.3, *Off-Diagonal Cost*) to take the shape of the respective alignment path into consideration. Especially in the case of musical automata, where performances differ considerably in terms of additional ornamentation, relying on DTW can result in relatively low alignment costs even when melody notes of one piece are aligned to auxiliary notes of the other piece. However, the rhythmic discrepancy arising from such a scenario will be reflected in the alignment path as significant non-diagonal sections.

The Off-Diagonal cost is an adequate measure for this effect. We define it as the deviation of the actual alignment path from the optimal strictly linear path through the dissimilarity matrix. The best linear path represents an alignment under the constraint of a constant tempo change and no rhythmic deviations and is obtained by performing a line detection on the inverted dissimilarity matrix. To this end, the Hough transform, as known from image processing, is applied. For a detailed description of the Hough transform and on how to compute the Off-Diagonal cost see Section 5.2.3.

7.4.6. Data Merging

In summary, after the processing steps described so far, the pieces have been split into chunks, and two plausibility measures for the alignment between each pair of such chunks have been computed considering each possible transposition interval. To finally obtain a single similarity measure these pieces of information need to be integrated.

First, the two similarity measures need to be combined. The Relative Path cost describes the difference between feature vectors along the alignment path in comparison to a specific baseline influenced by changes in instrumentation or recording conditions between the two audio recordings. The Off-Diagonal cost, on the other hand, measures the severity of changes in rhythm or local tempo. Preliminary experiments have shown that simply taking the product of these two measures results in a meaningful aggregated matching cost.

Next, conflicts between evidence for transpositions of segments by different intervals need to be resolved. To this end, a majority vote is taken from the n most similar pairs of segments. To determine this number n, the lengths of valid alignment paths given different scenarios are considered. Let one piece be split into a chunks and the one it is compared to into b fragments, then the minimum number n_{min} of pairs needed to fully reflect an alignment path over the whole recordings along the diagonal is $\max(a, b)$. On the other hand, the maximum number n_{max} of pairs needed to cover an alignment path in the worst case – if it consists of many horizontal and vertical segments – is $a + b - 1$. Therefore a reasonable number of pairs of chunks to take into consideration is chosen as $n = \alpha \max(a, b)$ with $1 \leq \alpha < 2$, depending on how much deviation from the main diagonal an overall alignment path should be allowed to exhibit.

The main idea of splitting pieces into chunks was to compensate for major structural changes, e.g. the insertion or deletion of a prominent section of a piece. A large difference in performance time would be a cue for such a modification. Therefore, instead of forcing two whole pieces to be aligned, parts of the longer recordings are allowed to be left out. To this end, the length of the shorter recording was chosen to

be the determining factor, resulting in $\tilde{n} = \alpha \min(a, b)$. We set α to 1.5, to still give consideration to deviations in tempo. Experiments have shown that the number $\tilde{n}$ of pairs taken into account outperforms the original n.

Once the main transposition interval has been obtained via voting among these $\tilde{n}$ selected pairs of segments, deviating transposition intervals of individual pairs are penalized by multiplying the respective matching costs by a factor β. In the context of our data, $\beta = 2$ is sufficient to prevent low matching costs as a result of arbitrarily many different transpositions. The final matching cost is then obtained by averaging over the costs of the $\tilde{n}$ most similar pairs of fragments.

7.4.7. Experimental Results

The data corpus used for the evaluation comprises recordings of 89 mechanical music instruments, provided by the *Phonogram Archive* of the Austrian Academy of Sciences. About half of the pieces are played by flute clocks while the other half is performed by musical boxes. As described above, there are also significant differences in performance style and accompaniment. While some instruments only play the main melody notes, others make use of rich ornamentations. Also, the sounds of the instruments are wildly different.

Amongst the test data are three pairs of recordings pertaining to the same underlying piece (all of them performed by music boxes). They comprise (several) themes from Auber's opera *Fra Diavolo*, Mendelssohn's oratorio *Elias*, and Haydn's oratorio *The Creation*, respectively. These are the 'cover versions' we wish to discover in the experiment.

In analogy to the MIREX audio cover song detection task, each of the 3×2 test recordings is in turn used as the query file. The ranking of the corresponding recording amongst the 88 remaining candidates is then examined and given in Table 7.1. Although there is only one perfect match, we are satisfied with the results, given the nature of the data. In comparison to popular music, a different version of a piece is not only played on a different instrument, but, as can be presumed from the durations also given in Table 7.1, there are significant differences in which subset of the underlying piece is performed at all.

Although the proposed algorithm is not precise enough for the fully automatic identification of matching recordings in a music collection as difficult as the one at hand, it clearly outperforms a 'standard' method which does not include the Off-Diagonal cost measure and a split-and-merge approach. Looking at the mean rank of the correspond-

Query Piece	**Duration**		**Proposed**		**Standard**	
	Ver. 1	Ver. 2	Ver. 1	Ver. 2	Ver. 1	Ver. 2
Fra Diavolo	2:13	3:25	1	3	5	8
Elias	0:58	1:53	4	3	12	6
The Creation	0:58	1:55	5	9	8	11

Table 7.1.: Rank of the corresponding version of the same piece within a list of 88 candidates, i.e, rank of version 2 when the query was version 1 and the other way around. 'Standard' refers to a DTW-based matching algorithm without our two extensions *split-and-merge* and *Off-Diagonal cost* (but with *Relative Path cost*, which was already proposed by [Turetsky and Ellis, 2003]).

ing version of the 6 query recordings, our system achieves a value of 4.2 in comparison to 8.3. Taking into account that state-of-the-art cover version systems are not able to automatically achieve a perfect classification in an unconstrained setting, we consider the result of this first study promising (cf. [Niedermayer et al., 2011b]).

7.5. Other Applications of Audio Alignment

For the purpose of completeness, two additional applications of audio alignment from the literature are briefly described. The first one, desoloing, requires an accurate alignment such that notes played by the soloist are removed while minimizing the damage to the sound of other instruments. For the second one, music retrieval, in contrast, the objective has to be to obtain alignments at a certain quality level as fast as possible, such that (relatively) large databases can be searched.

7.5.1. Desoloing

[Han and Raphael, 2010] describe a system for the suppression of the contribution of a soloist to the recording of a classical piece. The remaining orchestral accompaniment can then, by means of phase vocoding, be used in an automatic accompaniment system, where the obtained residual audio is played back according to another soloist's tempo.

The proposed method for the desoloing step works in several passes. In the first one, the timing of each note played by the solo instrument is obtained applying Audio-to-Score Alignment. Then, those notes are removed in the frequency domain using specific masks. Those masks consist of components which remove the steady partials present during the sustain phase of a note and an additional component removing the transient at the note's onset.

The problematic aspect of this approach is that the algorithm removes more than the actual contribution of the soloist. This is due to overlapping harmonics, i.e., the consequence that partials of notes played by accompanying instruments which have the same or a very similar frequency are removed as well. To overcome missing partials in the accompaniment, [Han and Raphael, 2010] propose a reconstruction method based on phase estimation using Kalman smoothing, projection from measured partials of a note onto the missing ones, and phase-locked modulation. The authors report a significant number of partials to be repaired using this approach in an evaluation on an excerpt of a length of 45 seconds from a piano concerto.

7.5.2. Query-by-Humming and Music Retrieval

In an alignment based music retrieval system, the objective is to find the best matching piece of music within a database, given a query sample. Assuming that the query is monophonic, such as in the query-by-humming scenario (see [Dannenberg et al., 2007], [Kapur et al., 2004], or [Song et al., 2002], for example), its fragments can correspond to different voices of a polyphonic piece. Standard string matching algorithms are, therefore, not an adequate solution to this problem.

[Pardo and Sanghi, 2005], for example, propose a music retrieval system based on audio alignment in the symbolic domain (cf. Section 4.3). There, the query sequence is aligned to each score in the database and the corresponding score with the lowest matching cost is returned. The polyphonic alignment is performed by a dynamic programming algorithm which allows for sections of the monophonic query to be aligned to different voices of the target.

In comparison the our alignment system working in the symbolic domain (see Section 4.3), [Pardo and Sanghi, 2005] assume a perfect transcription of the query. Based on this constraint, they report a significant improvement on their (small) test set over a reference algorithm which does not allow for sections of the query to correspond to different voices of the target. With a modeled probability that the query changes between two voices of 0.5 at each individual note, the vast majority of all test runs returned the correct target at the first rank, while this was only the case for about one third of the queries processed by the reference system. This significant improvement illustrates the potential of alignment techniques in the domain of music retrieval.

8. Conclusion

8.1. Summary

In this thesis we have described an Audio-to-Score Alignment system which was developed to obtain accurate note onset annotations from arbitrary recordings of piano pieces of which a score is known and available to the system. To this end we have performed detailed investigations into three audio features including one (Pitch Activation) that we have initially proposed for alignment tasks in [Niedermayer, 2009a]. Since all of these features are computed in the spectral domain, numerous transforms of an audio signal into a time-frequency representation were discussed. Here, consistent with Heisenberg's uncertainty principle, the time-resolution cannot be increased without reducing the frequency resolution and inversely. In search for an adequate trade-off, we investigated into several transforms and found the short-time Fourier transform and the constant Q transform to be the ones best suited for our system.

Based on an initial Audio-to-Score Alignment performed by Dynamic Time Warping we introduced several optimizations and refinement strategies. To obtain this initial alignment at a fine temporal resolution while, at the same time, keeping computational costs low, a two-scale approach was used. A first alignment path was computed at a rough temporal feature resolution with an additional Itakura parallelogram constraint. Based upon this first estimate, a refined alignment was computed where only paths within a certain corridor centered around the initial path were considered.

The shortcoming of this method as well as most stat-of-the-art systems is that individual chord notes cannot be resolved but are assigned a uniform timestamp instead. To overcome this drawback we introduced a two-pass post-processing approach, where, in the first step, anchor notes are extracted for which an accurate onset estimated can be obtained at a high level of confidence. To this end, our Pitch Activation feature is used. Then, following a Divide & Conquer approach, the notes in between or notated concurrently to these anchors are refined.

An extensive evaluation was performed on a set of 13 Mozart sonatas comprising more than 100.000 notes. It was shown that each of the proposed refinement steps significantly improves the degree to which our objective – to minimize the number of aligned notes a human annotator would need to correct – is achieved. For our final system this number amounts to 52.41% of all notes.

In addition we presented an Audio-to-Score Alignment system working in the symbolic domain based upon a quasi-transcription obtained from our Pitch Activation features (Section 4.3). Also, we presented a number of applications of Audio-to-Score Alignment based data acquisition in the domain of computational musicology. As a "side-product" demonstrating the general adaptability of the techniques described here, we developed a version detection system for historical musical automata and a music player which can automatically recognize the structure of a piece.

8.2. Discussion

While, as mentioned above, the alignment accuracy of each individual piece benefits from all our proposed refinement strategies and the overall result is promising, there is a number of open questions. First, classical piano music is a narrow field. While the restriction towards the classical genre is, up to a certain extent, justified by the availability of score material corresponding to audio recordings of musical performances, considering piano music only is a considerable constraint.

A second issue is robustness. While other authors report significant improvements of the robustness and, at the same time, the accuracy of DTW alignments when also using onset based features in addition to Chroma vectors, we could not confirm those findings in our experiments. We found that large numbers of notes which the pianist inserted in his performance caused the alignment based on a mixture of Chroma vectors and onset based descriptors to fail.

Also, at the current state of our research, we only extract note onset timings. Although this information is essential for the estimation of other note parameters such as duration or relative loudness and, therefore, a reasonable measurement task to start with, performing an onset extraction only is not satisfactory in the long term. Problems that will have to be faced are that even when, in the case of the piano, the exact key and pedal movements are known, one does not know where the offset of a note is exactly. The note might have decayed before the corresponding key is released. On the other hand, the relative loudness of individual chord notes is an issue which has been widely neglected in recent MIR research. Our preliminary experiments, however, showed that

relying on the energy content of spectrogram bins corresponding to a certain note is very fragile when processing mixtures of three or more notes.

Despite these limitations of our proposed system, we consider our results an achievement. We want to remind the reader of the difficulty of the task of accurate Audio-to-Score Alignment considering the large number of the performer's deviations from the score. More than 4,000 additional notes have been inserted during the performance amounting to almost 4% of the number of score notes. However, such variations or playing errors are a natural phenomenon an Audio-to-Score Alignment system has to be robust to. On our evaluation corpus, this is the case for our proposed alignment system.

8.3. Future Developments

From the above considerations we have identified three main areas for future work.

- In this thesis we mainly focused on increasing the alignment accuracy of those notes for which the initial onset estimate is accurate within a certain range. The refinement methods working within a fixed search window can inherently not correct the timings of those notes where the actual onset lies outside the search window. Therefore, an objective is to concentrate on those 0.68% of all notes which are aligned with a time deviation of more than one second. Given the nature of the data including a relatively large number of deviations of the performances from the scores, this is a challenging task. One possible approach would be to explicitly detect such deviations and to react accordingly.
  Robustness is also an issue when applying the alignment system to piano pieces by composers different from Mozart. While our system does not overfit the data in the sense that it implements composer specific performance models, it is reasonable to assume that pieces from the romantic era such as the works of Chopin are more difficult to process due to more prominent pedal usage and freedom concerning tempo variations in comparison to pieces of the classical era. Here, a medium term objective is to prepare an evaluation set comparable to the Mozart corpus used within this thesis.

- A major limitation of our current system is the focus on piano music. While current methods are not powerful enough to detect each individual note in pieces with a much higher degree of polyphony than possible on a piano, i.e., in pieces played by a whole orchestra, it is desirable to have a system which can automatically extract annotations of performances by soloists playing arbitrary instruments or

by small ensembles, such as string quartets. This requires the detection of *soft onsets* such as produced by strings or wind instruments where one cannot focus on recognizing transient attack phases.

- A third important objective for future developments is to also extract other note parameters in addition to the onset time. An obvious issue is the detection of note offsets, i.e., the duration of notes. This is problematic due to several aspects. While the onset of a note played on a piano is clearly defined as the time when the hammer hits the string resulting in a transient increase in spectral energy at the frequencies corresponding to the respective pitch, the offset can take various forms. On the one hand, a vibrating string can be silenced abruptly by releasing the key. On the other hand, a note can decay continuously while the key or the sustain pedal is pressed. In such cases, even for trained listeners it is hard to identify an accurate note offset in polyphonic pieces where masking effects and shared partials occur.
  Extracting the exact relative loudness of individual notes is an even more difficult problem. For instruments, such as the piano, where the loudness of a note cannot be further increased after the onset it is subjected to a continuous decay. The estimation of loudness, therefore, strongly depends on an accurate onset detection to be able to relate a measured energy to the respective degree of decay.

A. Performance Statistics

A.1. Tempo

piece	1^{st} movement		
	tempo	**time sig**	**bpm**
k.279	Allegro	4/4	113
k.280	Allegro assai	3/4	126
k.281	Allegro	2/4	67
k.282	Adagio	4/4	36
k.283	Allegro	3/4	135
k.284	Allegro	4/4	135
k.330	Allegro moderato	2/4	67
k.331	Andante grazioso (Allegro)	6/8 (4/4)	108
k.332	Allegro	3/4	161
k.333	Allegro	4/4	136
k.457	Molto allegro	4/4	166
k.475	Adagio	4/4	36
k.533	Allegro	2/2	162

Table A.1.: Tempo indication, time signature, and actual, performed tempo of the 1^{st} movements in [bpm], where for pieces in alla breve the quarter note is considered to be the beat

piece	**2^{nd} movement**		
	tempo	**time sig**	**bpm**
k.279	Andante	3/4	45
k.280	Adagio	6/8	90
k.281	Andante amoroso	3/8	81
k.282	Menuetto	3/4	123
k.283	Andante	4/4	37
k.284	Andante	3/4	56
k.330	Andante cantabile	3/4	45
k.331	Menuetto	3/4	129
k.332	Adagio	4/4	31
k.333	Andante cantabile	3/4	46
k.457	Adagio	4/4	30
k.475	Allegro (Andantino)	4/4 (3/4)	95
k.533	Andante	3/4	52

Table A.2.: Tempo indication, time signature, and actual, performed tempo of the 2^{nd} movements in [bpm]

piece	**3^{rd} movement**		
	tempo	**time sig**	**bpm**
k.279	Allegro	2/4	138
k.280	Presto	3/8	259
k.281	Allegro	2/2	164
k.282	Allegro	2/4	136
k.283	Presto	3/8	281
k.284	Andante	2/2	116
k.330	Allegro	2/4	83
k.331	Allegretto	2/4	131
k.332	Allegro assai	6/8	291
k.333	Allegretto grazioso	2/2	138
k.457	Allegro assai	3/4	209
k.475	Piu Allegro	3/4	47
k.533	Allegretto	2/2	122

Table A.3.: Tempo indication, time signature, and actual, performed tempo of the 3^{rd} movements in [bpm], where for pieces in alla breve the quarter note is considered to be the beat

A.2. Dynamics

	1^{st} movement				
	count	min	mean	(stddev)	max
1	16273	–	–	–	–
2	6622	0.000	22.406	(13.956)	87.037
3	2602	0.000	28.851	(14.000)	97.917
4	806	1.887	30.752	(14.258)	94.915
5	94	5.479	31.647	(12.544)	85.714
6	35	10.112	34.967	(10.828)	59.740
7	3	22.472	27.899	(6.332)	36.782
8	1	20.690	20.690	(0.000)	20.690

Table A.4.: Intra-chord dynamics deviations for the 1^{st} movements according to the degree of polyphony measured as the softest note's MIDI velocity relative to the loudest note's MIDI velocity not including grace notes

	2^{nd} movement				
	count	min	mean	(stddev)	max
1	7366	–	–	–	–
2	3476	0.000	28.847	(15.554)	84.906
3	1482	1.471	33.398	(13.979)	94.444
4	522	2.326	39.558	(13.786)	78.667
5	62	10.753	34.323	(11.046)	60.811
6	8	20.000	28.995	(6.832)	39.024
7	4	36.000	45.169	(6.487)	54.286

Table A.5.: Intra-chord dynamics deviations for the 2^{nd} movements according to the degree of polyphony measured as the softest note's MIDI velocity relative to the loudest note's MIDI velocity not including grace notes

	3^{rd} movement				
	count	min	mean	(stddev)	max
1	14721	–	–	–	–
2	6073	0.000	20.079	(13.887)	94.203
3	2479	0.000	27.357	(12.876)	88.136
4	574	4.545	30.679	(14.539)	92.857
5	159	8.235	28.592	(11.774)	58.537
6	20	8.696	32.305	(12.834)	63.415
7	0	–	–	–	–
8	15	14.286	27.477	(7.647)	50.000

Table A.6.: Intra-chord dynamics deviations for the 3^{rd} movements according to the degree of polyphony measured as the softest note's MIDI velocity relative to the loudest note's MIDI velocity not including grace notes

A.3. Micro-Timings

	1^{st} movement				
	count	min	mean	(stddev)	max
1	16273	–	–	–	–
2	6622	0.000	0.016	(0.019)	0.286
3	2602	0.000	0.018	(0.014)	0.236
4	806	0.001	0.028	(0.033)	0.292
5	94	0.005	0.065	(0.067)	0.227
6	35	0.005	0.113	(0.066)	0.215
7	3	0.010	0.014	(0.003)	0.017
8	1	0.009	0.009	(0.000)	0.009

Table A.7.: The 1^{st} movements' time spreads between the earliest and the latest note of a chord according to the respective degree of polyphony, disregarding ornamentations, i.e., grace notes and trills

	2^{nd} movement				
	count	min	mean	(stddev)	max
1	7366	–	–	–	–
2	3476	0.000	0.019	(0.025)	0.447
3	1482	0.000	0.026	(0.034)	0.514
4	522	0.002	0.034	(0.038)	0.435
5	62	0.007	0.065	(0.063)	0.330
6	8	0.016	0.022	(0.006)	0.030
7	4	0.019	0.030	(0.013)	0.051

Table A.8.: The 2^{nd} movements' time spreads between the earliest and the latest note of a chord according to the respective degree of polyphony, disregarding ornamentations, i.e., grace notes and trills

	3^{rd} movement				
	count	min	mean	(stddev)	max
1	14721	–	–	–	–
2	6073	0.000	0.014	(0.018)	0.322
3	2479	0.000	0.019	(0.017)	0.245
4	574	0.000	0.029	(0.042)	0.318
5	159	0.002	0.038	(0.049)	0.193
6	20	0.006	0.049	(0.054)	0.152
8	15	0.006	0.146	(0.145)	0.366

Table A.9.: The 3^{rd} movements' time spreads between the earliest and the latest note of a chord according to the respective degree of polyphony, disregarding ornamentations, i.e., grace notes and trills

	1^{st} movement				
	count	min	mean	(stddev)	max
1	46	–	–	–	–
2	304	0.000	0.028	(0.033)	0.161
3	143	0.001	0.055	(0.077)	0.471
4	45	0.054	0.208	(0.051)	0.391
5	36	0.125	0.281	(0.129)	0.529
6	11	0.090	0.289	(0.153)	0.511

Table A.10.: The 1^{st} movements' time spreads between the earliest and the latest note of a chord including ornamentation without its own dedicated timing in the score and timing differences between ornamentations with a notated timing and other notes having the same score time

	2^{nd} movement				
	count	min	mean	(stddev)	max
1	22	–	–	–	–
2	86	0.000	0.047	(0.055)	0.248
3	38	0.009	0.149	(0.118)	0.478
4	56	0.009	0.170	(0.157)	0.691
5	49	0.107	0.285	(0.130)	0.637
6	6	0.274	0.418	(0.131)	0.621
7	2	0.370	0.455	(0.085)	0.541

Table A.11.: The 2^{nd} movements' time spreads between the earliest and the latest note of a chord including ornamentation without its own dedicated timing in the score and timing differences between ornamentations with a notated timing and other notes having the same score time

	3^{rd} movement				
	count	min	mean	(stddev)	max
1	35	–	–	–	–
2	142	0.000	0.035	(0.047)	0.196
3	188	0.001	0.066	(0.060)	0.310
4	95	0.007	0.089	(0.055)	0.244
5	63	0.014	0.188	(0.116)	0.454
6	48	0.103	0.189	(0.069)	0.366
7	13	0.090	0.318	(0.157)	0.598
8	16	0.092	0.194	(0.210)	1.001

Table A.12.: The 3^{rd} movements' time spreads between the earliest and the latest note of a chord including ornamentation without its own dedicated timing in the score and timing differences between ornamentations with a notated timing and other notes having the same score time

B. Alignment Results

In the following the detailed evaluation results are presented for the entire corpus, the entire corpus clustered by movement numbers, and each individual piece. In addition to the performance of our complete Audio-to-Score Alignment system, we also give intermediary results for the initial DTW-based alignment step, the extracted anchors, and the remaining notes before and after their refinement. Given performance measures are the mean, standard deviation, minimum, and maximum of the time displacements as well as their absolute values (measures calculated on absolute errors are parenthesized) in milliseconds. In the following two lines the 5^{th} , 25^{th} , 50^{th} , 75^{th} , and 95^{th} percentiles are shown. Finally, the number of correctly aligned notes with respect to the 50 ms (as common in Onset Detection) and our proposed 10 ms tolerance range are given. For the individual movements of a piece, the time displacements are plotted over the note numbers. The respective histograms show the distributions of errors. In addition, the cumulative distribution functions of the timing errors are plotted. Since the maxima of these functions are known, the plots are scaled such that the entire available space is used.

Overall Result

(a) Two-scale DTW and Chroma vectors

avg-error				3.0 (80.0)
std-dev				204.3 (188.0)
min-error				-3386.5 (0.0)
max-error				4479.9 (4479.9)
-185.3 (1.9)	-23.7 (8.9)	-7.8 (20.2)	13.3 (62.5)	216.0 (362.7)
error<50ms				70.98%
error<10ms				28.44%

(b) Refined Anchor Notes

avg-error				6.2 (28.4)
std-dev				114.0 (110.5)
min-error				-3045.1 (0.0)
max-error				2753.3 (3045.1)
-12.4 (0.5)	-2.0 (2.6)	3.2 (6.1)	11.9 (15.0)	45.8 (99.9)
error<50ms				92.14%
error<10ms				65.41%

(c) Non-Anchor Notes before Refinement

avg-error				1.6 (99.5)
std-dev				234.4 (212.3)
min-error				-3386.5 (0.0)
max-error				4479.9 (4479.9)
-240.3 (2.6)	-32.3 (10.8)	-11.1 (25.2)	9.5 (89.7)	282.5 (435.5)
error<50ms				65.50%
error<10ms				22.73%

(d) Non-Anchor Notes after Refinement

avg-error				-1.0 (60.5)
std-dev				190.4 (180.6)
min-error				-3460.5 (0.0)
max-error				3776.3 (3776.3)
-142.2 (1.0)	-11.6 (5.2)	-4.1 (10.7)	8.0 (30.4)	113.9 (293.7)
error<50ms				82.01%
error<10ms				47.38%

(e) Complete System

avg-error				1.5 (50.5)
std-dev				169.9 (162.2)
min-error				-3460.5 (0.0)
max-error				3776.3 (3776.3)
-108.1 (0.8)	-8.8 (4.1)	-0.9 (9.3)	10.5 (24.1)	84.4 (232.4)
error<50ms				85.15%
error<10ms				52.41%

Table B.1.: Overall Alignment Results

1^{st} Movements

(a) Two-scale DTW and Chroma vectors

avg-error				6.6 (68.9)
std-dev				175.3 (161.4)
min-error				-2216.4 (0.0)
max-error				2943.7 (2943.7)
-143.4 (1.6)	-20.2 (8.0)	-6.8 (17.9)	12.8 (54.3)	200.4 (312.5)
error<50ms				73.57%
error<10ms				31.61%

(b) Refined Anchor Notes

avg-error				6.5 (25.1)
std-dev				97.6 (94.6)
min-error				-1816.3 (0.0)
max-error				2433.8 (2433.8)
-10.2 (0.5)	-1.8 (2.5)	3.3 (5.8)	11.7 (14.3)	44.4 (86.2)
error<50ms				92.58%
error<10ms				66.80%

(c) Non-Anchor Notes before Refinement

avg-error				6.6 (84.6)
std-dev				200.3 (181.7)
min-error				-2216.4 (0.0)
max-error				2943.7 (2943.7)
-185.5 (2.3)	-26.7 (9.7)	-9.6 (21.7)	9.9 (73.8)	257.0 (372.6)
error<50ms				68.66%
error<10ms				26.05%

(d) Non-Anchor Notes after Refinement

avg-error				-0.3 (52.8)
std-dev				165.4 (156.7)
min-error				-2179.9 (0.0)
max-error				2907.6 (2907.6)
-122.2 (1.0)	-10.9 (4.9)	-3.6 (10.2)	8.2 (29.3)	99.8 (234.9)
error<50ms				82.87%
error<10ms				49.21%

(e) Complete System

avg-error				2.0 (44.4)
std-dev				147.7 (140.9)
min-error				-2179.9 (0.0)
max-error				2907.6 (2907.6)
-100.4 (0.8)	-8.2 (3.9)	-0.6 (8.9)	10.4 (23.0)	76.2 (206.7)
error<50ms				85.83%
error<10ms				54.08%

Table B.2.: Overall Alignment Results: First Movements

2^{nd} Movements

(a) Two-scale DTW and Chroma vectors

avg-error				-4.5 (122.8)
std-dev				293.5 (266.6)
min-error				-3386.5 (0.0)
max-error				3508.5 (3508.5)
-354.9 (2.2)	-33.3 (10.7)	-9.6 (26.6)	14.8 (90.8)	360.1 (595.6)
error<50ms				65.29%
error<10ms				23.39%

(b) Refined Anchor Notes

avg-error				5.3 (37.0)
std-dev				158.2 (153.9)
min-error				-3045.1 (0.0)
max-error				2753.3 (3045.1)
-13.8 (0.5)	-2.1 (2.7)	3.2 (6.3)	12.2 (15.6)	46.0 (105.6)
error<50ms				92.17%
error<10ms				64.66%

(c) Non-Anchor Notes before Refinement

avg-error				-9.6 (162.5)
std-dev				344.1 (303.4)
min-error				-3386.5 (0.0)
max-error				3508.5 (3508.5)
-442.9 (3.9)	-49.9 (14.4)	-15.6 (37.6)	8.0 (178.8)	480.0 (724.9)
error<50ms				56.66%
error<10ms				15.80%

(d) Non-Anchor Notes after Refinement

avg-error				-6.8 (92.0)
std-dev				271.4 (255.4)
min-error				-3460.5 (0.0)
max-error				2647.2 (3460.5)
-314.3 (1.1)	-12.7 (5.5)	-4.8 (11.4)	7.6 (32.4)	205.4 (590.7)
error<50ms				80.56%
error<10ms				44.89%

(e) Complete System

avg-error				-2.2 (73.2)
std-dev				238.2 (226.7)
min-error				-3460.5 (0.0)
max-error				2753.3 (3460.5)
-184.9 (0.8)	-9.2 (4.3)	-0.9 (9.8)	10.9 (24.6)	101.5 (461.5)
error<50ms				84.44%
error<10ms				50.69%

Table B.3.: Overall Alignment Results: Second Movements

3^{rd} Movements

(a) Two-scale DTW and Chroma vectors

avg-error			3.1	(68.1)
std-dev			169.4	(155.1)
min-error			-2473.2	(0.0)
max-error			4479.9	(4479.9)
-150.1	-23.2	-8.1	13.0	192.8
(2.0)	(9.1)	(20.0)	(60.7)	(280.0)
error<50ms				71.37%
error<10ms				27.83%

(b) Refined Anchor Notes

avg-error			6.4	(26.7)
std-dev			95.7	(92.1)
min-error			-2386.1	(0.0)
max-error			2273.0	(2386.1)
-14.8	-2.4	3.1	12.1	47.4
(0.5)	(2.7)	(6.3)	(15.6)	(116.1)
error<50ms				91.64%
error<10ms				64.37%

(c) Non-Anchor Notes before Refinement

avg-error			2.1	(82.3)
std-dev			193.2	(174.8)
min-error			-2471.1	(0.0)
max-error			4479.9	(4479.9)
-184.4	-30.3	-11.0	9.7	234.3
(2.6)	(10.8)	(23.9)	(78.7)	(332.5)
error<50ms				66.75%
error<10ms				22.79%

(d) Non-Anchor Notes after Refinement

avg-error			1.3	(52.1)
std-dev			161.5	(152.9)
min-error			-2438.9	(0.0)
max-error			3776.3	(3776.3)
-121.3	-11.9	-4.4	7.9	109.1
(1.1)	(5.3)	(10.9)	(30.9)	(232.3)
error<50ms				81.86%
error<10ms				46.69%

(e) Complete System

avg-error			3.1	(44.4)
std-dev			144.4	(137.4)
min-error			-2438.9	(0.0)
max-error			3776.3	(3776.3)
-103.1	-9.1	-1.3	10.5	85.9
(0.8)	(4.3)	(9.6)	(24.8)	(210.1)
error<50ms				84.81%
error<10ms				51.56%

Table B.4.: Overall Alignment Results: Third Movements

k.279 – 1^{st} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			-0.2	(42.7)
std-dev			84.2	(72.5)
min-error			-499.3	(0.0)
max-error			663.0	(663.0)
-104.0 (1.6)	-18.1 (7.4)	-7.5 (14.8)	7.3 (41.3)	139.0 (183.6)
error<50ms				78.63%
error<10ms				35.11%

(b) Refined Anchor Notes

avg-error			6.1	(17.1)
std-dev			44.5	(41.6)
min-error			-343.4	(0.0)
max-error			539.1	(539.1)
-9.1 (0.5)	-2.0 (2.6)	3.1 (5.4)	10.9 (13.6)	39.8 (70.9)
error<50ms				93.21%
error<10ms				68.79%

(c) Non-Anchor Notes before Refinement

avg-error			-3.1	(50.9)
std-dev			96.7	(82.3)
min-error			-499.3	(0.0)
max-error			663.0	(663.0)
-132.9 (2.2)	-24.8 (8.6)	-9.9 (16.7)	2.9 (49.8)	177.4 (218.8)
error<50ms				75.01%
error<10ms				30.47%

(d) Non-Anchor Notes after Refinement

avg-error			-3.9	(32.0)
std-dev			77.5	(70.7)
min-error			-1639.2	(0.0)
max-error			552.1	(1639.2)
-97.3 (1.1)	-11.2 (5.3)	-4.8 (10.4)	6.5 (26.1)	82.7 (136.7)
error<50ms				85.12%
error<10ms				48.68%

(e) Complete System

avg-error			-0.3	(27.3)
std-dev			68.5	(62.8)
min-error			-1639.2	(0.0)
max-error			552.1	(1639.2)
-74.6 (0.8)	-8.4 (4.0)	-1.1 (8.7)	9.3 (21.7)	65.3 (126.0)
error<50ms				87.62%
error<10ms				54.91%

Table B.5.: Alignment Results k.279-1

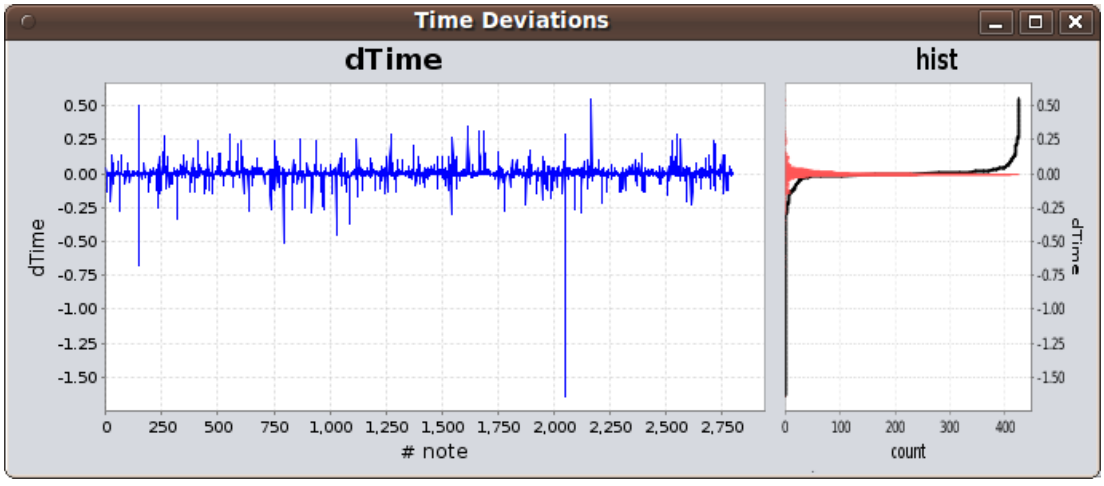


Figure B.1.: Time Deviations k.279-1

k.279 – 2nd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			-79.6	(173.1)
std-dev			365.8	(331.9)
min-error			-3205.7	(0.0)
max-error			1457.6	(3205.7)
-538.5 (3.9)	-72.1 (14.6)	-17.7 (36.6)	3.0 (217.6)	341.7 (723.3)
error<50ms				57.16%
error<10ms				15.32%

(b) Refined Anchor Notes

avg-error			0.2	(38.6)
std-dev			154.0	(149.1)
min-error			-2396.8	(0.0)
max-error			1257.6	(2396.8)
-44.3 (0.7)	-1.7 (2.9)	4.8 (7.6)	15.2 (19.9)	50.7 (162.2)
error<50ms				90.47%
error<10ms				58.17%

(c) Non-Anchor Notes before Refinement

avg-error			-109.8	(227.1)
std-dev			422.6	(372.9)
min-error			-3205.7	(0.0)
max-error			1457.6	(3205.7)
-715.1 (7.5)	-210.5 (19.0)	-25.9 (63.8)	-7.2 (323.5)	403.1 (810.7)
error<50ms				46.30%
error<10ms				8.49%

(d) Non-Anchor Notes after Refinement

avg-error			-60.9	(127.9)
std-dev			324.0	(303.8)
min-error			-3460.5	(0.0)
max-error			1250.2	(3460.5)
-486.4 (1.1)	-14.7 (5.5)	-5.4 (11.4)	5.7 (50.8)	253.9 (644.8)
error<50ms				74.79%
error<10ms				45.21%

(e) Complete System

avg-error			-42.3	(101.2)
std-dev			285.1	(269.9)
min-error			-3460.5	(0.0)
max-error			1257.6	(3460.5)
-431.2 (0.9)	-10.4 (4.8)	-1.8 (10.4)	10.4 (36.1)	97.8 (483.3)
error<50ms				79.34%
error<10ms				48.53%

Table B.6.: Alignment Results k.279-2

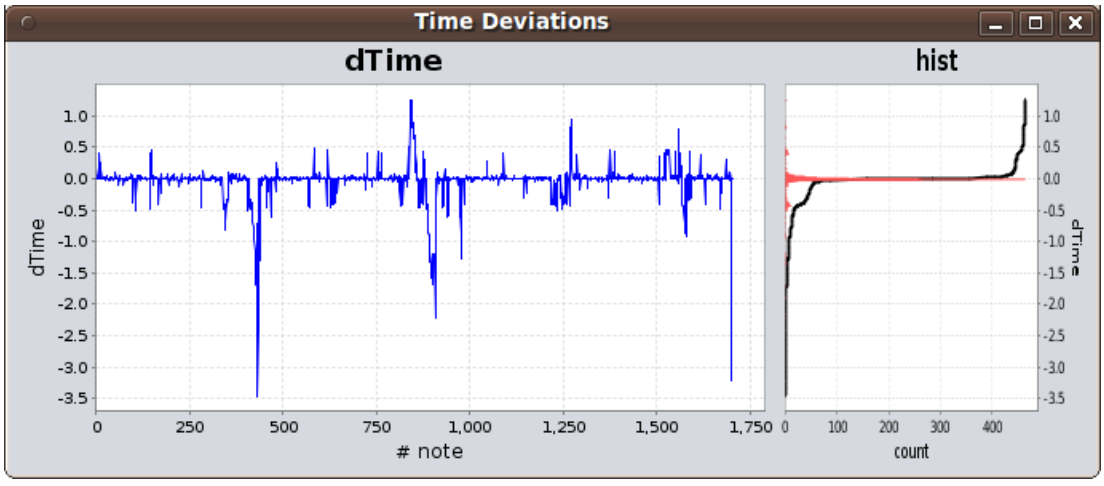


Figure B.2.: Time Deviations k.279-2

k.279 – 3rd Movement

(a) Two-scale DTW and Chroma vectors

avg-error				-2.6 (41.0)
std-dev				77.5 (65.8)
min-error				-579.7 (0.0)
max-error				576.7 (579.7)
-108.4 (1.4)	-18.5 (6.7)	-6.2 (15.2)	8.1 (42.7)	131.0 (192.3)
error<50ms				78.12%
error<10ms				37.08%

(b) Refined Anchor Notes

avg-error				7.4 (16.2)
std-dev				41.4 (38.8)
min-error				-489.3 (0.0)
max-error				490.5 (490.5)
-8.9 (0.5)	-0.7 (2.4)	4.1 (5.7)	11.1 (13.7)	46.9 (62.6)
error<50ms				93.71%
error<10ms				67.96%

(c) Non-Anchor Notes before Refinement

avg-error				-6.7 (48.6)
std-dev				87.8 (73.5)
min-error				-579.7 (0.0)
max-error				576.7 (579.7)
-129.4 (1.8)	-26.4 (8.1)	-9.3 (17.3)	3.8 (52.0)	170.7 (208.4)
error<50ms				74.28%
error<10ms				31.41%

(d) Non-Anchor Notes after Refinement

avg-error				-1.5 (27.6)
std-dev				57.1 (50.0)
min-error				-498.9 (0.0)
max-error				464.3 (498.9)
-86.7 (0.9)	-10.8 (4.8)	-3.9 (10.0)	7.8 (23.4)	69.4 (121.3)
error<50ms				86.29%
error<10ms				50.10%

(e) Complete System

avg-error				1.4 (24.2)
std-dev				53.0 (47.2)
min-error				-498.9 (0.0)
max-error				490.5 (498.9)
-66.1 (0.7)	-8.2 (3.9)	-0.3 (8.9)	9.9 (20.3)	57.6 (109.2)
error<50ms				88.54%
error<10ms				55.07%

Table B.7.: Alignment Results k.279-3

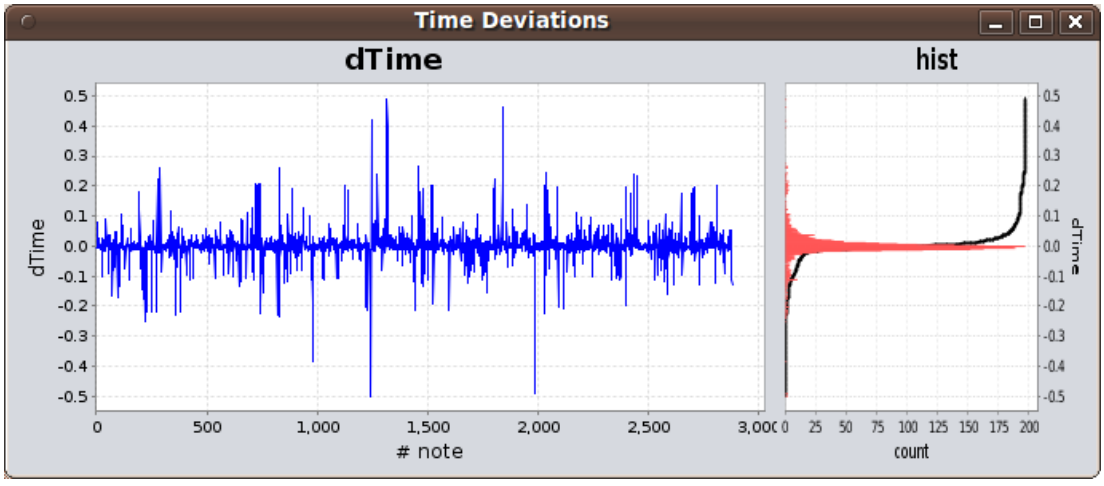


Figure B.3.: Time Deviations k.279-3

k.280 – 1^{st} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			-10.2	(52.7)
std-dev			111.0	(98.2)
min-error			-862.2	(0.0)
max-error			890.6	(890.6)
-175.5 (1.6)	-17.8 (6.8)	-6.1 (14.6)	9.6 (52.8)	109.5 (230.7)
error<50ms				74.11%
error<10ms				38.02%

(b) Refined Anchor Notes

avg-error			1.7	(20.4)
std-dev			61.2	(57.8)
min-error			-578.3	(0.0)
max-error			267.9	(578.3)
-8.1 (0.7)	-0.5 (2.6)	4.6 (6.1)	11.7 (14.1)	38.1 (76.9)
error<50ms				93.61%
error<10ms				66.86%

(c) Non-Anchor Notes before Refinement

avg-error			-17.1	(65.0)
std-dev			128.7	(112.4)
min-error			-862.2	(0.0)
max-error			890.6	(890.6)
-209.3 (2.1)	-27.9 (8.1)	-8.8 (18.4)	4.4 (67.7)	134.9 (317.1)
error<50ms				69.38%
error<10ms				32.02%

(d) Non-Anchor Notes after Refinement

avg-error			-4.8	(32.9)
std-dev			88.7	(82.5)
min-error			-767.1	(0.0)
max-error			1325.1	(1325.1)
-96.8 (0.7)	-9.4 (4.2)	-3.0 (8.6)	6.7 (22.5)	73.0 (154.2)
error<50ms				85.36%
error<10ms				56.99%

(e) Complete System

avg-error			-2.5	(28.8)
std-dev			80.6	(75.4)
min-error			-767.1	(0.0)
max-error			1325.1	(1325.1)
-69.1 (0.7)	-6.8 (3.5)	0.1 (7.8)	9.4 (18.4)	60.1 (135.7)
error<50ms				88.08%
error<10ms				60.02%

Table B.8.: Alignment Results k.280-1

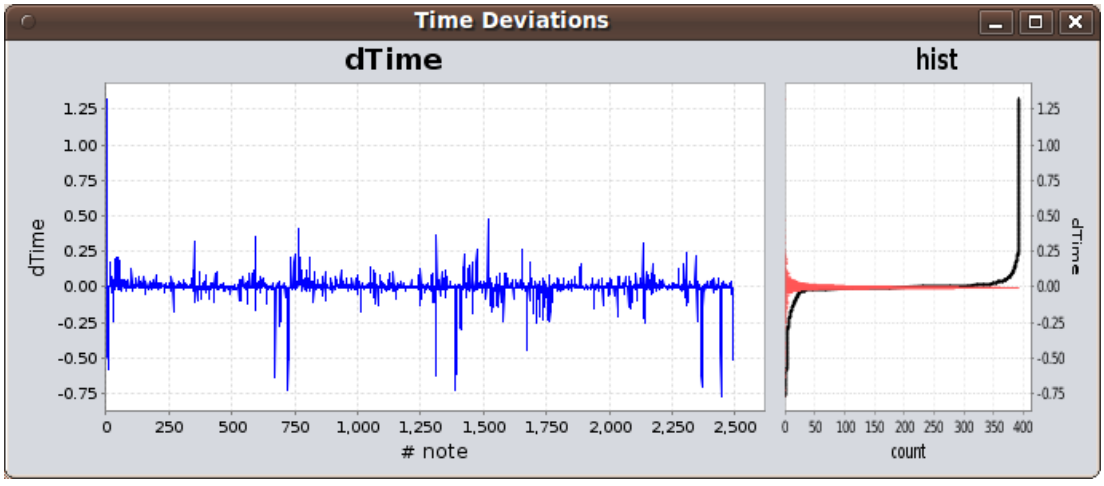


Figure B.4.: Time Deviations k.280-1

k.280 – 2^{nd} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			137.8	(237.2)
std-dev			490.4	(450.8)
min-error			-1257.0	(0.0)
max-error			3230.1	(3230.1)
-310.3 (2.0)	-26.5 (10.9)	-2.5 (37.4)	65.5 (225.0)	1150.4 (1159.7)
error<50ms				57.12%
error<10ms				23.20%

(b) Refined Anchor Notes

avg-error			22.3	(41.2)
std-dev			161.2	(157.4)
min-error			-834.8	(0.0)
max-error			1201.1	(1201.1)
-9.7 (0.6)	-2.4 (2.7)	2.9 (6.4)	12.7 (14.6)	45.8 (76.1)
error<50ms				93.76%
error<10ms				63.55%

(c) Non-Anchor Notes before Refinement

avg-error			203.1	(343.2)
std-dev			593.7	(525.3)
min-error			-1257.0	(0.0)
max-error			3230.1	(3230.1)
-422.0 (4.4)	-41.2 (19.0)	-7.8 (81.0)	312.4 (487.8)	1395.3 (1395.3)
error<50ms				43.27%
error<10ms				13.59%

(d) Non-Anchor Notes after Refinement

avg-error			93.9	(153.8)
std-dev			385.8	(366.0)
min-error			-1403.7	(0.0)
max-error			2061.2	(2061.2)
-44.9 (1.7)	-12.1 (6.7)	-4.6 (13.6)	21.0 (38.0)	1039.7 (1074.1)
error<50ms				78.36%
error<10ms				37.03%

(e) Complete System

avg-error			67.7	(112.8)
std-dev			324.2	(311.4)
min-error			-1403.7	(0.0)
max-error			2061.2	(2061.2)
-33.5 (1.0)	-8.8 (4.9)	0.1 (11.0)	16.5 (26.1)	796.2 (834.1)
error<50ms				83.92%
error<10ms				46.05%

Table B.9.: Alignment Results k.280-2

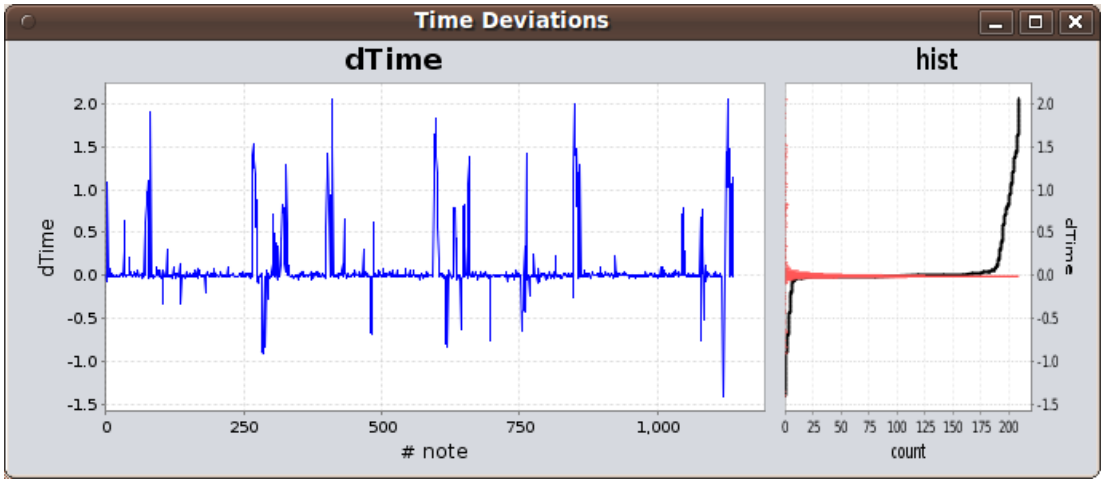


Figure B.5.: Time Deviations k.280-2

k.280 – 3rd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			14.1	(63.1)
std-dev			120.8	(103.9)
min-error			-722.0	(0.0)
max-error			894.3	(894.3)
-113.9 (2.0)	-16.6 (9.2)	-4.3 (21.0)	33.7 (70.2)	211.7 (266.3)
error<50ms				68.33%
error<10ms				27.86%

(b) Refined Anchor Notes

avg-error			6.2	(23.3)
std-dev			65.4	(61.4)
min-error			-594.2	(0.0)
max-error			516.3	(594.2)
-14.3 (0.5)	-2.8 (2.6)	2.3 (5.8)	10.2 (14.0)	64.3 (116.2)
error<50ms				91.44%
error<10ms				67.67%

(c) Non-Anchor Notes before Refinement

avg-error			16.8	(77.9)
std-dev			140.8	(118.6)
min-error			-722.0	(0.0)
max-error			894.3	(894.3)
-138.0 (2.6)	-21.7 (11.0)	-6.9 (26.2)	40.6 (97.0)	261.1 (343.2)
error<50ms				62.84%
error<10ms				22.27%

(d) Non-Anchor Notes after Refinement

avg-error			-1.5	(41.9)
std-dev			101.0	(91.9)
min-error			-1484.5	(0.0)
max-error			907.1	(1484.5)
-112.9 (1.3)	-11.3 (5.1)	-4.2 (11.0)	9.2 (35.1)	106.4 (211.3)
error<50ms				80.27%
error<10ms				46.53%

(e) Complete System

avg-error			1.4	(35.7)
std-dev			90.3	(83.0)
min-error			-1484.5	(0.0)
max-error			907.1	(1484.5)
-87.6 (0.8)	-8.4 (4.1)	-1.3 (9.2)	11.1 (25.9)	91.6 (202.2)
error<50ms				84.10%
error<10ms				52.56%

Table B.10.: Alignment Results k.280-3

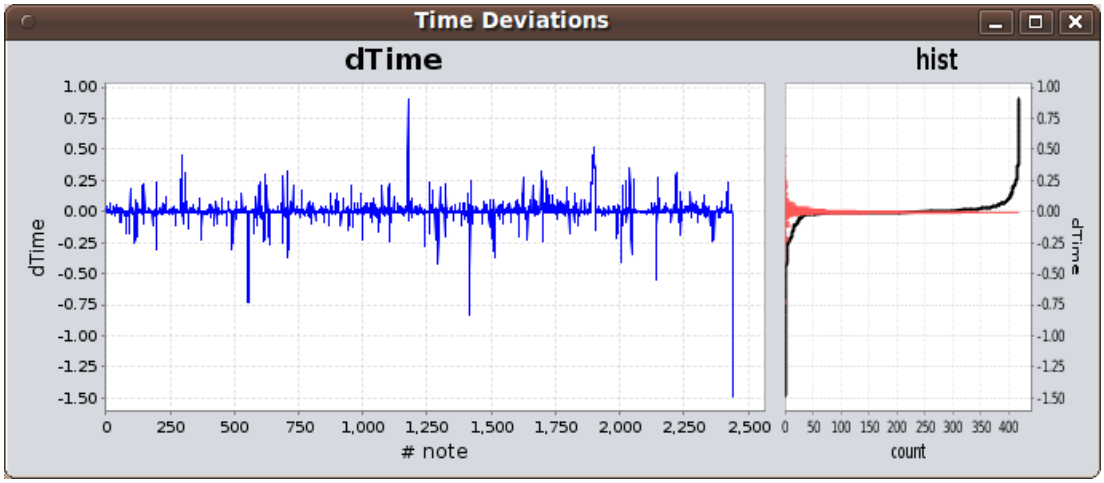


Figure B.6.: Time Deviations k.280-3

k.281 – 1^{st} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			-1.5	(63.6)
std-dev			130.5	(114.0)
min-error			-1086.3	(0.0)
max-error			1087.2	(1087.2)
-161.6 (2.1)	-25.5 (10.5)	-9.0 (22.8)	17.6 (65.5)	170.0 (235.6)
error<50ms				69.94%
error<10ms				23.53%

(b) Refined Anchor Notes

avg-error			0.1	(25.7)
std-dev			79.8	(75.5)
min-error			-586.3	(0.1)
max-error			1019.3	(1019.3)
-48.0 (0.5)	-3.2 (2.4)	1.3 (5.4)	7.9 (11.8)	44.3 (184.1)
error<50ms				90.30%
error<10ms				72.21%

(c) Non-Anchor Notes before Refinement

avg-error			-3.5	(74.1)
std-dev			145.6	(125.4)
min-error			-1086.3	(0.0)
max-error			1087.2	(1087.2)
-187.5 (3.1)	-33.6 (12.4)	-12.3 (26.9)	12.8 (82.1)	190.2 (264.8)
error<50ms				65.41%
error<10ms				19.42%

(d) Non-Anchor Notes after Refinement

avg-error			-12.7	(43.0)
std-dev			112.7	(104.9)
min-error			-1400.0	(0.0)
max-error			1343.4	(1400.0)
-141.4 (1.5)	-13.1 (6.4)	-6.8 (11.8)	5.0 (30.5)	71.7 (208.0)
error<50ms				82.55%
error<10ms				42.23%

(e) Complete System

avg-error			-8.8	(38.1)
std-dev			104.3	(97.5)
min-error			-1400.0	(0.0)
max-error			1343.4	(1400.0)
-127.6 (0.9)	-11.0 (4.5)	-3.2 (9.9)	7.0 (22.4)	68.8 (200.3)
error<50ms				84.59%
error<10ms				50.45%

Table B.11.: Alignment Results k.281-1

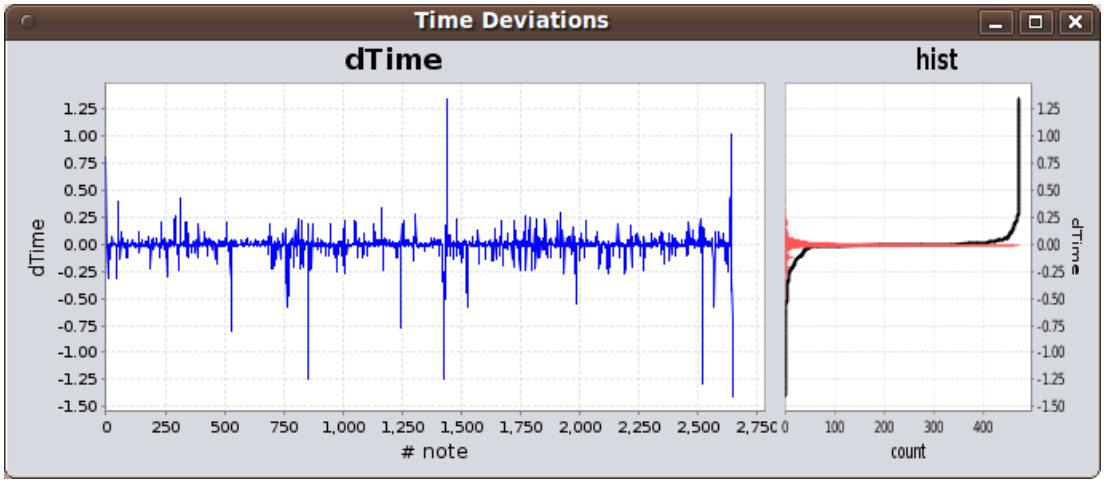


Figure B.7.: Time Deviations k.281-1

k.281 – 2nd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			4.6	(105.4)
std-dev			227.8	(202.0)
min-error			-1328.6	(0.0)
max-error			1521.4	(1521.4)
-290.5 (2.0)	-28.0 (9.0)	-4.8 (27.9)	27.8 (78.2)	366.0 (551.0)
error<50ms				64.65%
error<10ms				27.07%

(b) Refined Anchor Notes

avg-error			0.2	(31.6)
std-dev			123.5	(119.4)
min-error			-1304.2	(0.0)
max-error			828.5	(1304.2)
-7.6 (0.5)	-0.4 (2.9)	5.4 (7.1)	14.2 (17.4)	42.4 (58.5)
error<50ms				93.59%
error<10ms				62.90%

(c) Non-Anchor Notes before Refinement

avg-error			6.3	(143.6)
std-dev			273.4	(232.8)
min-error			-1328.6	(0.0)
max-error			1521.4	(1521.4)
-384.7 (3.6)	-41.0 (12.4)	-10.6 (39.3)	27.4 (169.2)	481.0 (668.1)
error<50ms				55.87%
error<10ms				19.27%

(d) Non-Anchor Notes after Refinement

avg-error			-14.5	(59.8)
std-dev			168.7	(158.4)
min-error			-1316.4	(0.0)
max-error			743.5	(1316.4)
-221.4 (1.0)	-11.8 (5.6)	-5.7 (10.3)	5.2 (24.7)	77.8 (412.7)
error<50ms				82.67%
error<10ms				48.22%

(e) Complete System

avg-error			-8.2	(49.2)
std-dev			153.0	(145.1)
min-error			-1316.4	(0.0)
max-error			828.5	(1316.4)
-67.7 (0.8)	-8.4 (4.4)	-0.1 (9.4)	11.1 (21.4)	67.9 (264.3)
error<50ms				86.79%
error<10ms				52.76%

Table B.12.: Alignment Results k.281-2

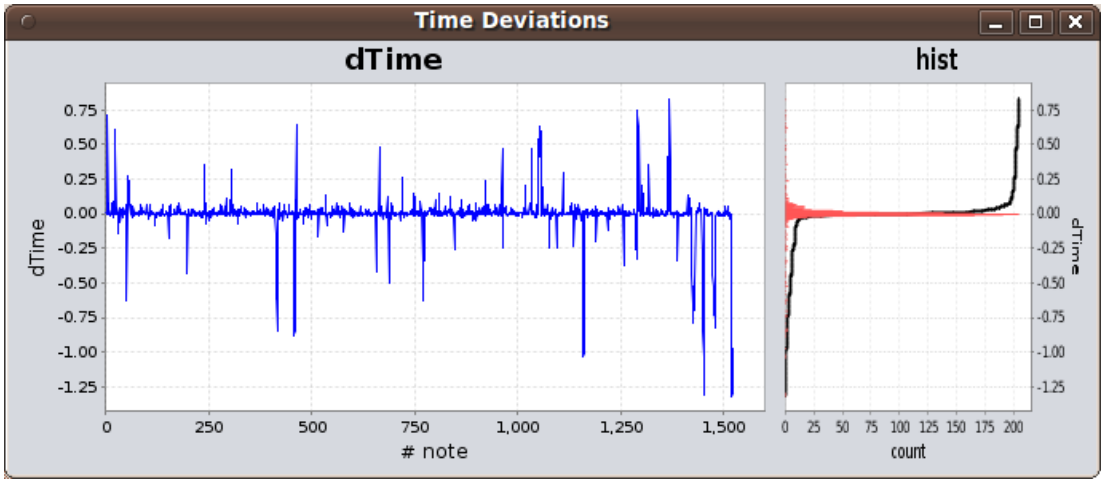


Figure B.8.: Time Deviations k.281-2

k.281 – 3rd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			8.3	(72.0)
std-dev			163.2	(146.7)
min-error			-1343.9	(0.0)
max-error			1368.8	(1368.8)
-150.3 (2.1)	-20.0 (9.1)	-7.3 (18.7)	15.7 (61.6)	269.9 (326.6)
error<50ms				71.42%
error<10ms				27.64%

(b) Refined Anchor Notes

avg-error			2.8	(26.3)
std-dev			98.7	(95.2)
min-error			-1153.8	(0.1)
max-error			1112.3	(1153.8)
-17.2 (0.7)	-1.8 (2.8)	3.5 (5.8)	10.8 (13.9)	37.3 (97.1)
error<50ms				92.63%
error<10ms				67.59%

(c) Non-Anchor Notes before Refinement

avg-error			9.6	(89.0)
std-dev			185.5	(163.1)
min-error			-1343.9	(0.0)
max-error			1368.8	(1368.8)
-179.2 (2.7)	-27.1 (10.9)	-10.8 (23.3)	12.4 (89.9)	319.6 (356.5)
error<50ms				66.07%
error<10ms				21.71%

(d) Non-Anchor Notes after Refinement

avg-error			-8.3	(49.1)
std-dev			144.6	(136.2)
min-error			-2148.3	(0.0)
max-error			1144.3	(2148.3)
-136.3 (1.1)	-11.7 (5.1)	-5.5 (10.4)	3.6 (28.6)	80.4 (227.4)
error<50ms				83.07%
error<10ms				48.80%

(e) Complete System

avg-error			-4.5	(41.9)
std-dev			131.7	(124.9)
min-error			-2148.3	(0.0)
max-error			1144.3	(2148.3)
-107.9 (0.8)	-9.2 (4.0)	-2.0 (9.0)	8.1 (21.2)	64.9 (191.7)
error<50ms				86.16%
error<10ms				54.16%

Table B.13.: Alignment Results k.281-3

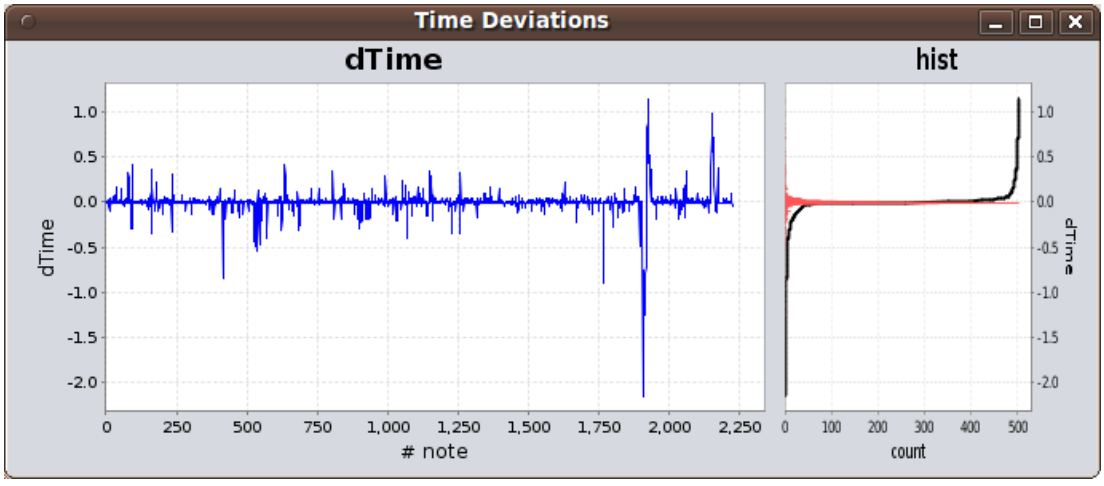


Figure B.9.: Time Deviations k.281-3

k.282 – 1^{st} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			-6.4	(97.3)
std-dev			206.4	(182.1)
min-error			-1382.5	(0.0)
max-error			1623.6	(1623.6)
-309.6 (2.8)	-35.7 (12.2)	-11.4 (27.8)	15.5 (81.5)	280.9 (425.7)
error<50ms				66.02%
error<10ms				19.58%

(b) Refined Anchor Notes

avg-error			-2.8	(27.9)
std-dev			102.7	(98.9)
min-error			-758.9	(0.0)
max-error			1231.1	(1231.1)
-32.8 (0.6)	-3.9 (2.6)	1.5 (5.8)	10.0 (13.9)	34.9 (86.1)
error<50ms				92.99%
error<10ms				67.04%

(c) Non-Anchor Notes before Refinement

avg-error			-9.8	(126.1)
std-dev			241.5	(206.3)
min-error			-1382.5	(0.1)
max-error			1623.6	(1623.6)
-353.6 (4.3)	-51.8 (15.6)	-16.7 (37.2)	7.4 (162.7)	340.2 (516.3)
error<50ms				57.22%
error<10ms				12.79%

(d) Non-Anchor Notes after Refinement

avg-error			-0.5	(77.5)
std-dev			220.7	(206.6)
min-error			-1558.2	(0.0)
max-error			2164.8	(2164.8)
-266.5 (1.7)	-14.9 (7.4)	-7.9 (13.7)	3.7 (40.7)	175.9 (407.9)
error<50ms				78.06%
error<10ms				35.04%

(e) Complete System

avg-error			-1.0	(61.2)
std-dev			189.9	(179.8)
min-error			-1558.2	(0.0)
max-error			2164.8	(2164.8)
-206.4 (1.0)	-12.3 (5.3)	-4.1 (11.4)	8.6 (28.2)	86.1 (374.2)
error<50ms				83.06%
error<10ms				45.07%

Table B.14.: Alignment Results k.282-1

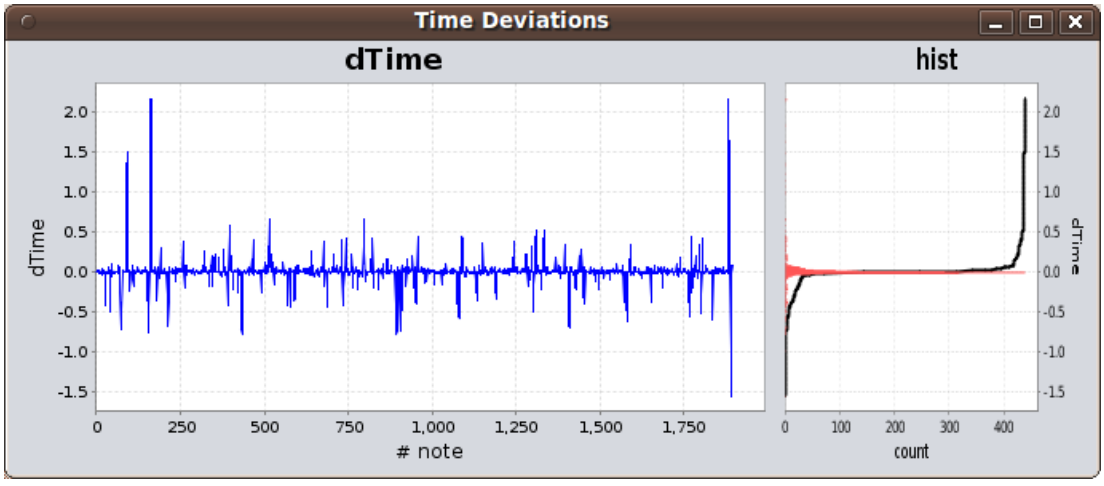


Figure B.10.: Time Deviations k.282-1

k.282 – 2^{nd} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			-4.2	(68.0)
std-dev			127.5	(107.9)
min-error			-887.1	(0.0)
max-error			543.7	(887.1)
-180.7 (2.6)	-27.0 (12.2)	-12.3 (22.0)	11.3 (68.6)	235.5 (324.3)
error<50ms				68.71%
error<10ms				20.21%

(b) Refined Anchor Notes

avg-error			2.6	(23.0)
std-dev			71.4	(67.6)
min-error			-498.7	(0.1)
max-error			558.1	(558.1)
-36.3 (0.7)	-6.4 (2.7)	-1.8 (6.0)	5.3 (11.2)	40.5 (116.2)
error<50ms				90.95%
error<10ms				72.40%

(c) Non-Anchor Notes before Refinement

avg-error			-10.2	(88.5)
std-dev			152.1	(124.1)
min-error			-887.1	(0.0)
max-error			543.7	(887.1)
-249.6 (4.3)	-38.5 (14.5)	-16.0 (27.7)	3.0 (126.2)	310.1 (357.4)
error<50ms				61.61%
error<10ms				13.67%

(d) Non-Anchor Notes after Refinement

avg-error			-0.0	(39.5)
std-dev			98.8	(90.6)
min-error			-559.3	(0.0)
max-error			752.7	(752.7)
-74.7 (1.8)	-15.8 (8.7)	-10.2 (13.8)	-0.0 (23.0)	77.5 (149.7)
error<50ms				85.49%
error<10ms				30.71%

(e) Complete System

avg-error			1.5	(33.4)
std-dev			89.2	(82.7)
min-error			-559.3	(0.0)
max-error			752.7	(752.7)
-67.8 (1.1)	-13.2 (5.3)	-5.9 (11.1)	3.5 (19.3)	64.8 (126.9)
error<50ms				87.43%
error<10ms				45.92%

Table B.15.: Alignment Results k.282-2

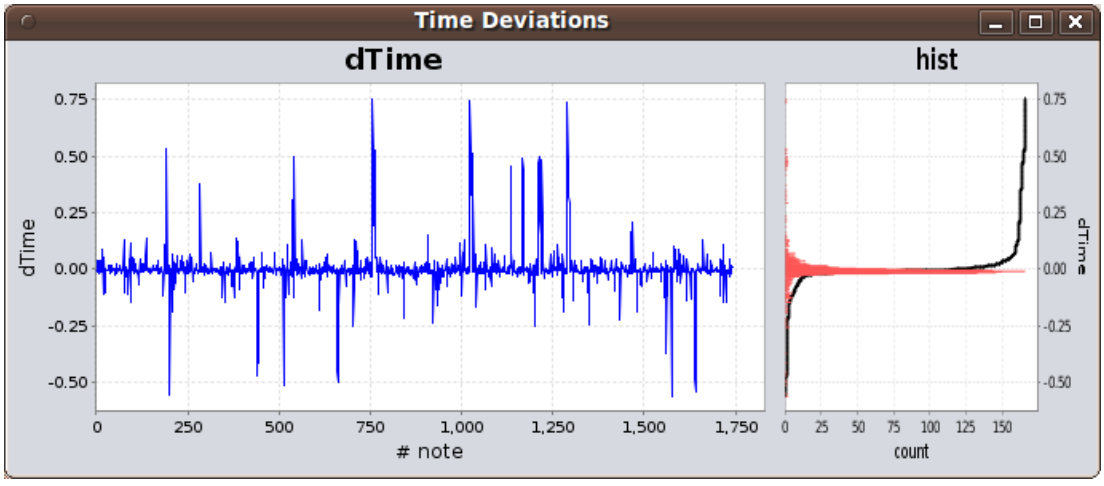


Figure B.11.: Time Deviations k.282-2

k.282 – 3rd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			5.4	(46.2)
std-dev			77.1	(61.9)
min-error			-421.9	(0.0)
max-error			433.6	(433.6)
-103.7	-18.8	-6.3	21.8	153.4
(2.2)	(9.6)	(19.5)	(56.9)	(185.8)
error<50ms				72.79%
error<10ms				26.21%

(b) Refined Anchor Notes

avg-error			2.8	(23.0)
std-dev			59.6	(55.1)
min-error			-280.2	(0.0)
max-error			433.8	(433.8)
-14.7	-4.3	1.0	10.3	38.8
(0.5)	(2.8)	(5.9)	(13.5)	(201.6)
error<50ms				92.17%
error<10ms				67.25%

(c) Non-Anchor Notes before Refinement

avg-error			5.6	(53.2)
std-dev			84.9	(66.5)
min-error			-421.9	(0.0)
max-error			433.6	(433.6)
-115.9	-23.0	-10.1	24.0	175.5
(2.6)	(12.2)	(23.5)	(67.7)	(205.2)
error<50ms				68.00%
error<10ms				20.16%

(d) Non-Anchor Notes after Refinement

avg-error			-14.6	(41.2)
std-dev			122.4	(116.2)
min-error			-1717.2	(0.0)
max-error			592.2	(1717.2)
-133.5	-15.1	-8.5	4.7	63.7
(1.9)	(7.8)	(13.4)	(29.7)	(194.5)
error<50ms				84.90%
error<10ms				35.02%

(e) Complete System

avg-error			-8.0	(34.9)
std-dev			104.7	(99.0)
min-error			-1717.2	(0.0)
max-error			592.2	(1717.2)
-100.4	-12.1	-3.9	8.5	63.0
(1.0)	(5.1)	(11.0)	(23.1)	(197.1)
error<50ms				87.26%
error<10ms				46.16%

Table B.16.: Alignment Results k.282-3

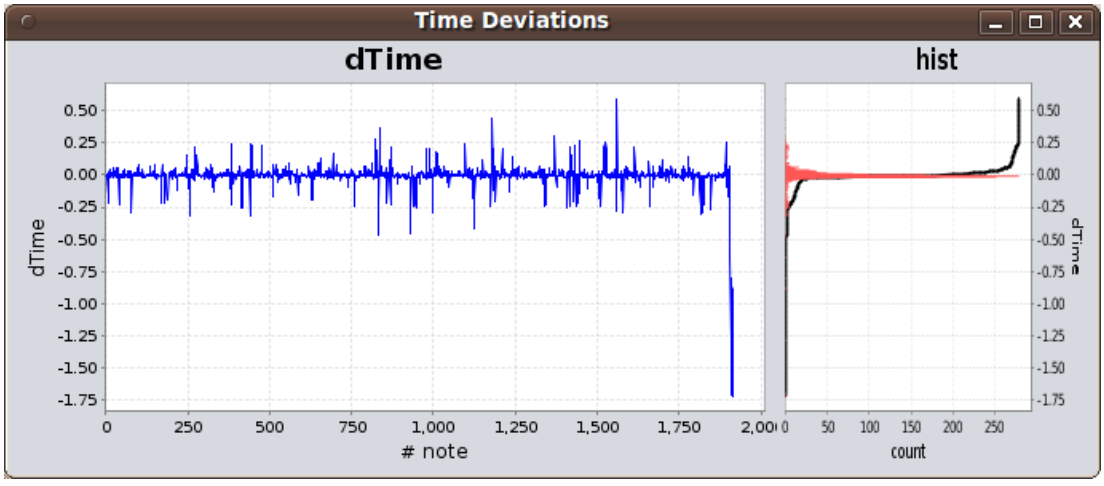


Figure B.12.: Time Deviations k.282-3

k.283 – 1^{st} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			0.8	(45.7)
std-dev			97.0	(85.6)
min-error			-653.5	(0.0)
max-error			923.6	(923.6)
-94.6	-18.6	-8.1	6.0	158.4
(1.5)	(7.7)	(15.6)	(39.6)	(212.0)
error<50ms				79.09%
error<10ms				33.63%

(b) Refined Anchor Notes

avg-error			5.1	(17.4)
std-dev			50.4	(47.6)
min-error			-435.6	(0.0)
max-error			552.2	(552.2)
-11.3	-2.9	1.5	8.4	40.0
(0.4)	(2.2)	(5.1)	(11.9)	(71.5)
error<50ms				93.21%
error<10ms				71.26%

(c) Non-Anchor Notes before Refinement

avg-error			-0.4	(54.8)
std-dev			110.5	(95.9)
min-error			-653.5	(0.0)
max-error			923.6	(923.6)
-108.6	-22.7	-10.4	4.4	194.9
(2.1)	(9.3)	(18.1)	(50.6)	(244.1)
error<50ms				74.82%
error<10ms				27.40%

(d) Non-Anchor Notes after Refinement

avg-error			-4.1	(32.9)
std-dev			84.7	(78.1)
min-error			-1701.2	(0.0)
max-error			763.8	(1701.2)
-100.3	-11.3	-4.9	5.6	67.3
(1.0)	(5.0)	(10.0)	(23.6)	(166.2)
error<50ms				85.85%
error<10ms				50.23%

(e) Complete System

avg-error			-1.3	(28.4)
std-dev			76.4	(71.0)
min-error			-1701.2	(0.0)
max-error			763.8	(1701.2)
-74.1	-9.0	-2.0	7.3	57.6
(0.7)	(3.8)	(8.4)	(20.0)	(126.5)
error<50ms				88.14%
error<10ms				55.99%

Table B.17.: Alignment Results k.283-1

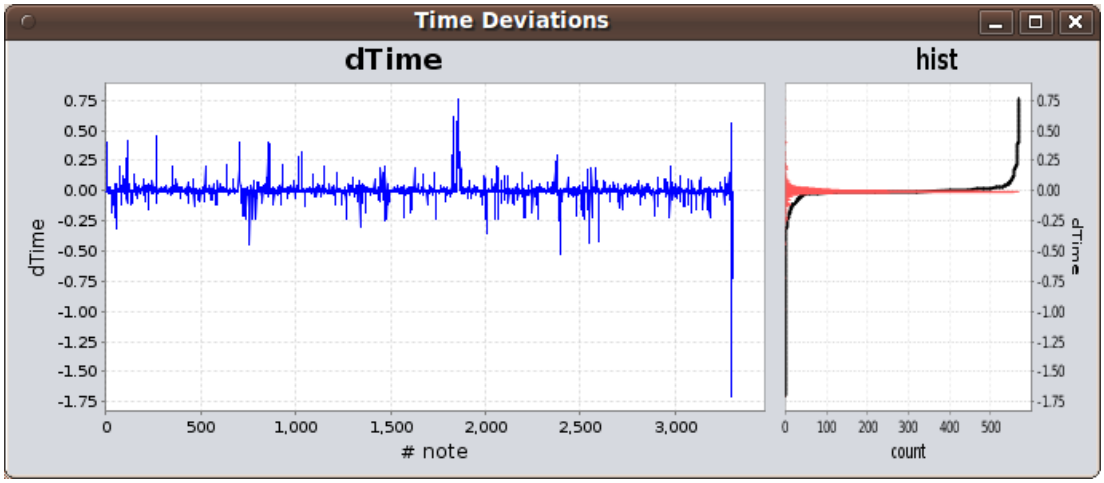


Figure B.13.: Time Deviations k.283-1

k.283 – 2^{nd} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			-28.6	(75.2)
std-dev			180.9	(166.9)
min-error			-1864.7	(0.0)
max-error			1500.6	(1864.7)
-264.5 (2.7)	-37.0 (9.7)	-13.5 (22.0)	-3.3 (53.5)	119.0 (354.9)
error<50ms				73.37%
error<10ms				25.73%

(b) Refined Anchor Notes

avg-error			2.3	(18.2)
std-dev			66.4	(63.9)
min-error			-583.1	(0.0)
max-error			847.9	(847.9)
-8.9 (0.4)	-1.2 (2.2)	3.4 (5.2)	10.8 (12.5)	38.9 (44.5)
error<50ms				95.80%
error<10ms				68.27%

(c) Non-Anchor Notes before Refinement

avg-error			-40.1	(97.3)
std-dev			213.5	(194.2)
min-error			-1864.7	(0.1)
max-error			1500.6	(1864.7)
-325.2 (4.6)	-48.7 (12.8)	-19.4 (28.1)	-7.9 (76.2)	187.1 (400.0)
error<50ms				67.08%
error<10ms				17.93%

(d) Non-Anchor Notes after Refinement

avg-error			-16.3	(68.6)
std-dev			206.0	(194.9)
min-error			-1853.1	(0.0)
max-error			1661.8	(1853.1)
-296.3 (0.8)	-10.2 (4.2)	-4.6 (8.4)	3.0 (19.0)	73.8 (467.3)
error<50ms				85.39%
error<10ms				58.89%

(e) Complete System

avg-error			-10.1	(52.5)
std-dev			174.0	(166.2)
min-error			-1853.1	(0.0)
max-error			1661.8	(1853.1)
-186.9 (0.6)	-8.2 (3.6)	-1.5 (7.6)	6.6 (17.3)	49.9 (360.1)
error<50ms				88.64%
error<10ms				61.31%

Table B.18.: Alignment Results k.283-2

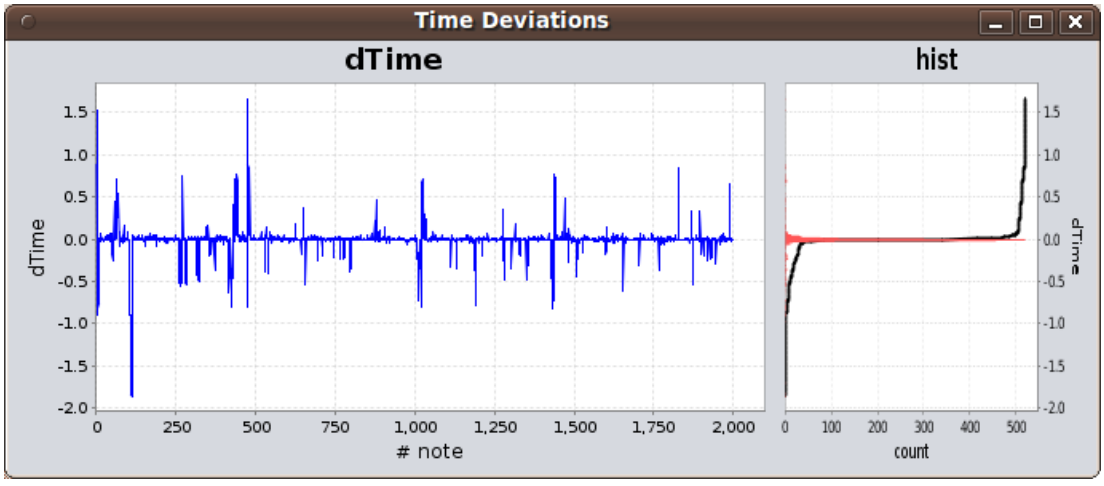


Figure B.14.: Time Deviations k.283-2

k.283 – 3rd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			8.5	(53.2)
std-dev			162.7	(154.0)
min-error			-503.6	(0.0)
max-error			2383.2	(2383.2)
-114.9 (1.8)	-18.3 (7.9)	-7.7 (15.2)	8.7 (45.0)	152.5 (187.5)
error<50ms				76.53%
error<10ms				32.82%

(b) Refined Anchor Notes

avg-error			11.6	(24.2)
std-dev			97.3	(95.0)
min-error			-396.8	(0.0)
max-error			1460.9	(1460.9)
-10.1 (0.5)	-2.0 (2.6)	3.1 (5.8)	11.3 (14.2)	42.4 (101.8)
error<50ms				92.20%
error<10ms				67.52%

(c) Non-Anchor Notes before Refinement

avg-error			7.3	(62.4)
std-dev			184.3	(173.6)
min-error			-503.6	(0.0)
max-error			2383.2	(2383.2)
-132.6 (2.1)	-23.4 (8.9)	-9.9 (17.6)	5.2 (56.7)	179.0 (213.0)
error<50ms				72.98%
error<10ms				28.23%

(d) Non-Anchor Notes after Refinement

avg-error			11.4	(45.5)
std-dev			177.4	(171.8)
min-error			-462.3	(0.0)
max-error			2383.2	(2383.2)
-93.6 (1.2)	-11.9 (5.5)	-5.1 (10.8)	7.1 (27.1)	93.8 (194.7)
error<50ms				83.93%
error<10ms				46.62%

(e) Complete System

avg-error			11.5	(39.2)
std-dev			157.4	(152.9)
min-error			-462.3	(0.0)
max-error			2383.2	(2383.2)
-79.5 (0.8)	-9.3 (4.3)	-1.8 (9.4)	9.5 (22.5)	76.0 (174.4)
error<50ms				86.42%
error<10ms				52.25%

Table B.19.: Alignment Results k.283-3

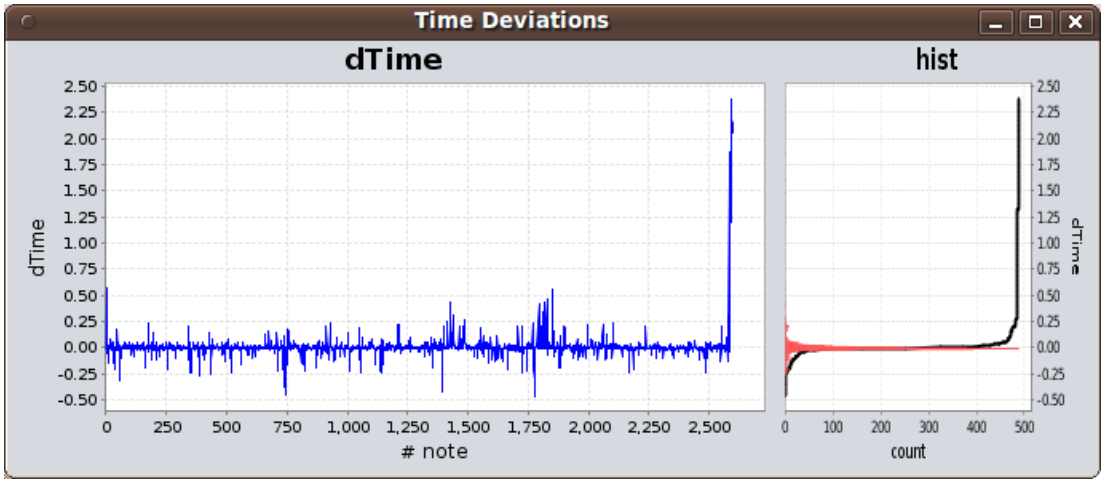


Figure B.15.: Time Deviations k.283-3

k.284 – 1^{st} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			1.2	(51.4)
std-dev			109.6	(96.8)
min-error			-906.3	(0.0)
max-error			978.0	(978.0)
-122.8	-17.6	-5.6	11.4	163.3
(1.3)	(7.1)	(15.6)	(50.3)	(227.6)
error<50ms				74.97%
error<10ms				35.32%

(b) Refined Anchor Notes

avg-error			6.1	(26.0)
std-dev			81.4	(77.4)
min-error			-861.3	(0.0)
max-error			972.2	(972.2)
-10.3	-1.9	2.7	11.5	51.5
(0.4)	(2.3)	(5.3)	(14.7)	(173.3)
error<50ms				91.05%
error<10ms				67.45%

(c) Non-Anchor Notes before Refinement

avg-error			-0.7	(58.2)
std-dev			118.3	(103.0)
min-error			-906.3	(0.0)
max-error			924.9	(924.9)
-139.1	-22.0	-7.8	10.0	199.5
(1.6)	(8.2)	(18.0)	(60.6)	(262.9)
error<50ms				71.79%
error<10ms				30.82%

(d) Non-Anchor Notes after Refinement

avg-error			-3.0	(43.3)
std-dev			99.1	(89.2)
min-error			-1027.9	(0.0)
max-error			896.5	(1027.9)
-134.5	-10.6	-2.9	9.0	107.7
(1.1)	(4.8)	(10.0)	(33.5)	(214.9)
error<50ms				81.03%
error<10ms				49.94%

(e) Complete System

avg-error			-0.3	(38.7)
std-dev			94.7	(86.4)
min-error			-1027.9	(0.0)
max-error			972.2	(1027.9)
-118.3	-8.0	-0.6	10.4	95.7
(0.8)	(3.8)	(8.9)	(26.4)	(209.9)
error<50ms				83.78%
error<10ms				54.14%

Table B.20.: Alignment Results k.284-1

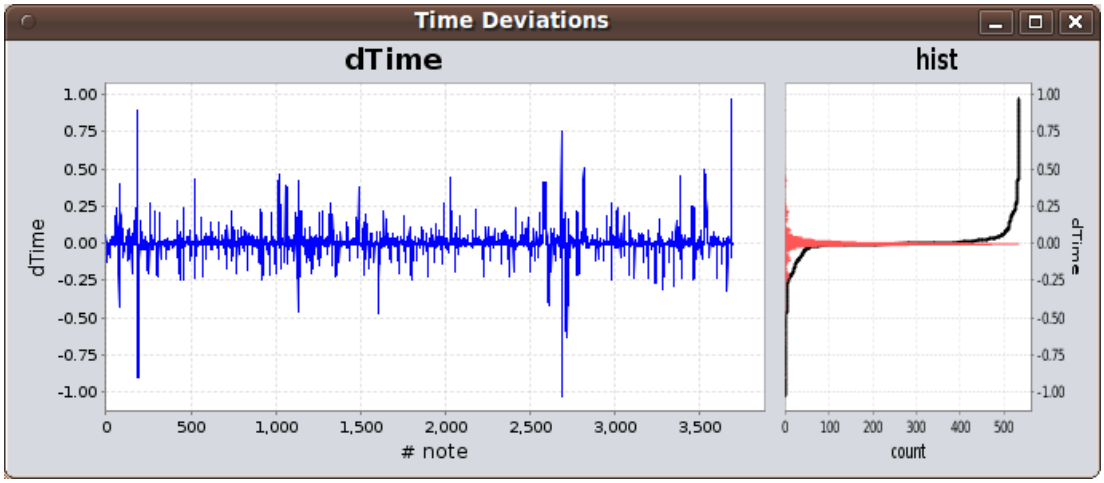


Figure B.16.: Time Deviations k.284-1

k.284 – 2nd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			55.0	(130.3)
std-dev			330.4	(308.6)
min-error			-929.6	(0.0)
max-error			3508.5	(3508.5)
-214.2 (2.4)	-28.6 (10.7)	-9.9 (25.2)	16.5 (93.1)	570.9 (594.4)
error<50ms				67.04%
error<10ms				23.26%

(b) Refined Anchor Notes

avg-error			20.7	(44.3)
std-dev			206.4	(202.7)
min-error			-925.4	(0.0)
max-error			2753.3	(2753.3)
-16.0 (0.6)	-2.5 (2.5)	2.2 (5.8)	12.5 (17.3)	52.2 (161.3)
error<50ms				91.39%
error<10ms				64.77%

(c) Non-Anchor Notes before Refinement

avg-error			75.2	(169.8)
std-dev			374.7	(342.3)
min-error			-929.6	(0.1)
max-error			3508.5	(3508.5)
-263.2 (4.2)	-36.4 (14.1)	-12.9 (36.0)	34.3 (187.3)	777.8 (801.9)
error<50ms				58.25%
error<10ms				13.98%

(d) Non-Anchor Notes after Refinement

avg-error			29.6	(90.4)
std-dev			279.4	(266.0)
min-error			-2117.5	(0.0)
max-error			2647.2	(2647.2)
-139.8 (1.2)	-10.1 (5.6)	-2.8 (11.0)	15.4 (32.5)	332.8 (523.7)
error<50ms				81.49%
error<10ms				46.78%

(e) Complete System

avg-error			27.0	(75.1)
std-dev			256.9	(247.2)
min-error			-2117.5	(0.0)
max-error			2753.3	(2753.3)
-90.3 (0.9)	-7.9 (4.2)	-0.1 (9.6)	15.3 (28.0)	232.4 (458.3)
error<50ms				84.72%
error<10ms				51.69%

Table B.21.: Alignment Results k.284-2

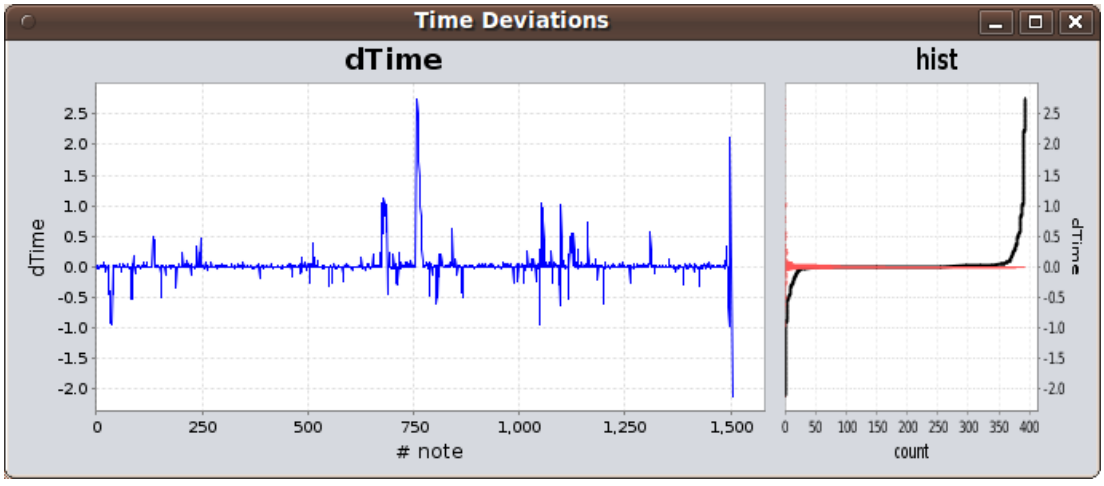


Figure B.17.: Time Deviations k.284-2

k.284 – 3rd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			-19.4	(72.3)
std-dev			160.0	(144.0)
min-error			-1394.5	(0.0)
max-error			2439.1	(2439.1)
-203.4 (2.7)	-38.3 (12.2)	-14.7 (25.5)	2.8 (69.6)	145.2 (286.4)
error<50ms				67.95%
error<10ms				19.70%

(b) Refined Anchor Notes

avg-error			2.3	(31.8)
std-dev			107.6	(102.8)
min-error			-1355.8	(0.0)
max-error			2273.0	(2273.0)
-85.4 (0.6)	-2.6 (2.9)	3.2 (7.0)	14.0 (19.5)	47.9 (186.6)
error<50ms				89.68%
error<10ms				60.18%

(c) Non-Anchor Notes before Refinement

avg-error			-25.9	(85.1)
std-dev			177.0	(157.3)
min-error			-1394.5	(0.0)
max-error			2439.1	(2439.1)
-240.2 (3.7)	-50.0 (14.7)	-18.8 (30.3)	-2.8 (89.9)	184.5 (321.9)
error<50ms				63.24%
error<10ms				14.69%

(d) Non-Anchor Notes after Refinement

avg-error			-11.5	(53.3)
std-dev			147.5	(138.0)
min-error			-1740.2	(0.0)
max-error			2096.6	(2096.6)
-178.1 (1.0)	-12.1 (5.3)	-4.3 (10.7)	7.6 (33.4)	101.7 (250.0)
error<50ms				80.08%
error<10ms				47.42%

(e) Complete System

avg-error			-7.2	(47.1)
std-dev			137.1	(129.0)
min-error			-1740.2	(0.0)
max-error			2273.0	(2273.0)
-160.8 (0.8)	-9.5 (4.4)	-1.3 (9.9)	10.4 (28.3)	82.7 (226.0)
error<50ms				82.90%
error<10ms				50.63%

Table B.22.: Alignment Results k.284-3

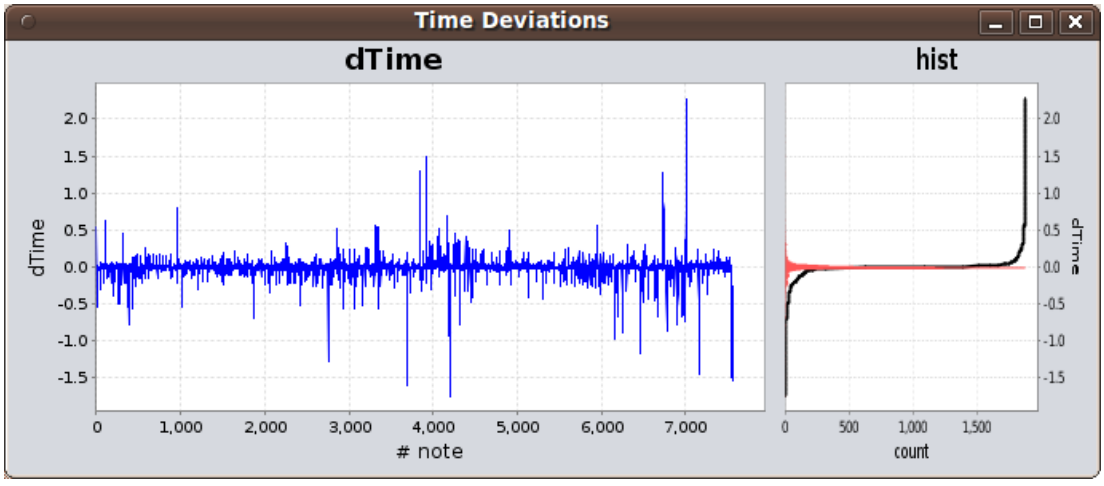


Figure B.18.: Time Deviations k.284-3

k.330 – 1^{st} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			-8.1	(45.8)
std-dev			98.7	(87.8)
min-error			-812.7	(0.0)
max-error			924.5	(924.5)
-115.1 (2.1)	-21.9 (8.4)	-10.1 (16.1)	3.0 (38.9)	102.6 (207.2)
error<50ms				79.91%
error<10ms				30.73%

(b) Refined Anchor Notes

avg-error			3.7	(12.6)
std-dev			37.8	(35.8)
min-error			-679.8	(0.0)
max-error			264.5	(679.8)
-8.8 (0.5)	-3.2 (2.4)	1.3 (5.0)	7.8 (9.5)	34.8 (41.1)
error<50ms				95.47%
error<10ms				76.92%

(c) Non-Anchor Notes before Refinement

avg-error			-11.5	(56.8)
std-dev			114.6	(100.2)
min-error			-812.7	(0.0)
max-error			924.5	(924.5)
-153.9 (3.5)	-27.7 (10.6)	-12.6 (19.5)	-0.7 (49.1)	163.4 (252.4)
error<50ms				75.44%
error<10ms				22.84%

(d) Non-Anchor Notes after Refinement

avg-error			-7.7	(26.8)
std-dev			72.5	(67.8)
min-error			-1636.9	(0.0)
max-error			474.9	(1636.9)
-86.2 (1.0)	-11.4 (4.8)	-5.4 (9.2)	3.0 (20.7)	46.3 (110.1)
error<50ms				88.96%
error<10ms				53.01%

(e) Complete System

avg-error			-4.2	(22.6)
std-dev			64.3	(60.3)
min-error			-1636.9	(0.0)
max-error			474.9	(1636.9)
-48.1 (0.8)	-9.0 (3.7)	-2.9 (7.9)	5.5 (16.9)	44.1 (96.3)
error<50ms				90.98%
error<10ms				59.81%

Table B.23.: Alignment Results k.330-1

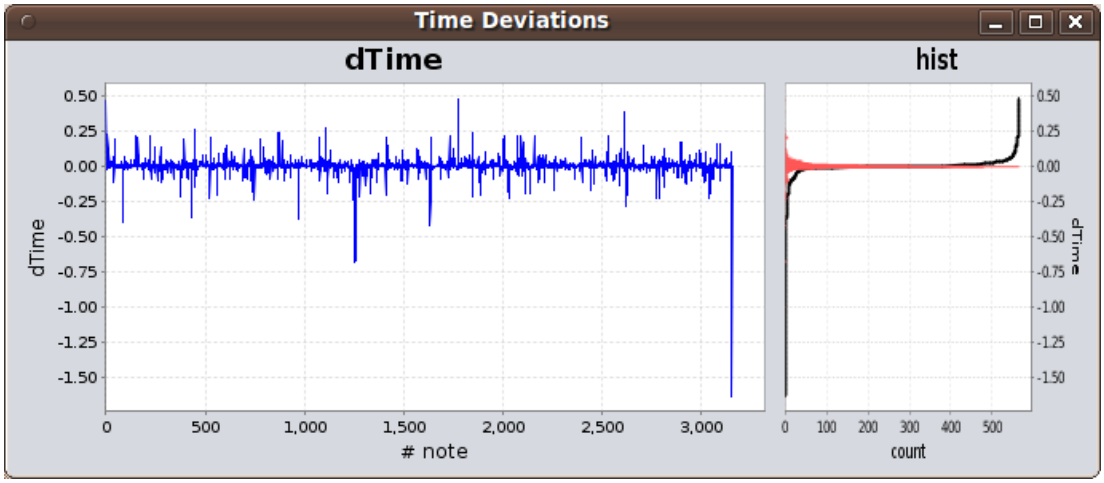


Figure B.19.: Time Deviations k.330-1

k.330 – 2nd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			16.7	(217.2)
std-dev			409.4	(347.4)
min-error			-2154.9	(0.1)
max-error			2351.5	(2351.5)
-599.5 (4.5)	-55.2 (18.5)	-16.4 (48.7)	31.9 (306.3)	676.1 (901.8)
error<50ms				50.76%
error<10ms				11.47%

(b) Refined Anchor Notes

avg-error			5.6	(81.5)
std-dev			245.6	(231.8)
min-error			-1855.8	(0.0)
max-error			1365.2	(1855.8)
-294.3 (0.6)	-1.1 (2.6)	4.0 (7.9)	16.3 (26.5)	129.1 (603.3)
error<50ms				84.96%
error<10ms				56.55%

(c) Non-Anchor Notes before Refinement

avg-error			21.9	(257.3)
std-dev			449.5	(369.3)
min-error			-2154.9	(0.5)
max-error			2351.5	(2351.5)
-661.5 (7.8)	-80.8 (23.1)	-21.3 (74.6)	47.1 (372.2)	781.0 (991.8)
error<50ms				43.12%
error<10ms				7.57%

(d) Non-Anchor Notes after Refinement

avg-error			3.4	(163.9)
std-dev			379.9	(342.7)
min-error			-2160.7	(0.0)
max-error			1520.9	(2160.7)
-637.1 (1.1)	-11.9 (5.0)	-3.0 (11.2)	9.7 (60.4)	684.3 (950.2)
error<50ms				73.47%
error<10ms				45.85%

(e) Complete System

avg-error			4.5	(144.1)
std-dev			351.9	(321.0)
min-error			-2160.7	(0.0)
max-error			1520.9	(2160.7)
-608.0 (0.9)	-9.9 (4.5)	-0.4 (11.1)	14.1 (45.6)	655.4 (827.9)
error<50ms				75.96%
error<10ms				46.50%

Table B.24.: Alignment Results k.330-2

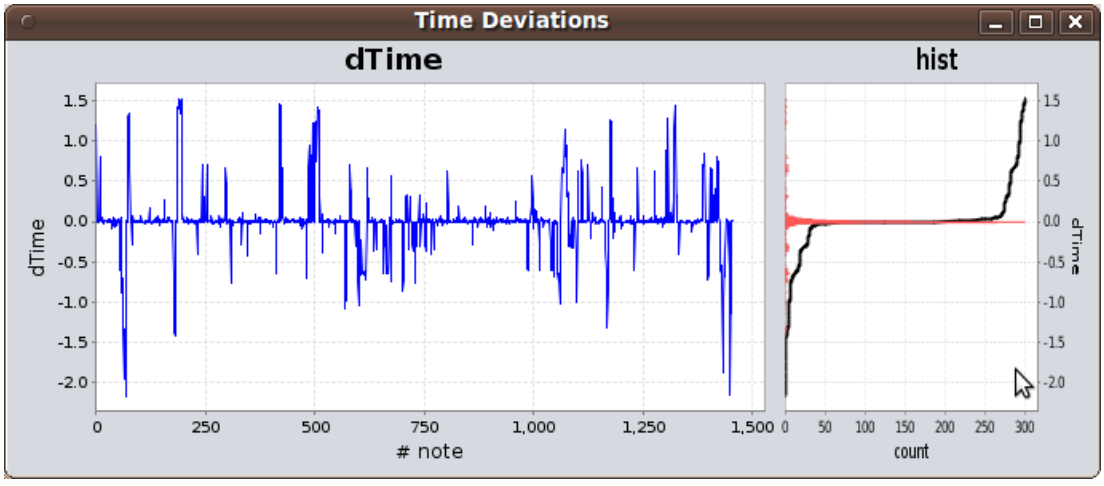


Figure B.20.: Time Deviations k.330-2

k.330 – 3rd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			-6.9	(37.7)
std-dev			71.4	(61.0)
min-error			-400.6	(0.0)
max-error			1055.8	(1055.8)
-104.9	-22.4	-9.3	3.4	82.3
(1.9)	(7.9)	(16.2)	(42.0)	(137.2)
error<50ms				78.81%
error<10ms				32.16%

(b) Refined Anchor Notes

avg-error			4.5	(15.9)
std-dev			41.0	(38.1)
min-error			-349.8	(0.0)
max-error			386.0	(386.0)
-9.4	-2.8	2.0	10.6	43.0
(0.4)	(2.4)	(5.7)	(13.2)	(59.0)
error<50ms				94.03%
error<10ms				69.24%

(c) Non-Anchor Notes before Refinement

avg-error			-11.1	(43.4)
std-dev			80.6	(68.8)
min-error			-400.6	(0.0)
max-error			1055.8	(1055.8)
-121.1	-29.7	-11.6	-0.6	91.1
(2.4)	(9.1)	(18.1)	(48.3)	(160.7)
error<50ms				75.96%
error<10ms				28.04%

(d) Non-Anchor Notes after Refinement

avg-error			-4.0	(23.4)
std-dev			61.9	(57.5)
min-error			-1064.8	(0.0)
max-error			1067.4	(1067.4)
-42.1	-11.6	-5.7	2.7	43.7
(1.0)	(5.0)	(9.7)	(18.7)	(95.6)
error<50ms				91.40%
error<10ms				51.37%

(e) Complete System

avg-error			-1.2	(21.1)
std-dev			56.1	(52.0)
min-error			-1064.8	(0.0)
max-error			1067.4	(1067.4)
-31.7	-9.2	-2.4	6.6	43.0
(0.7)	(3.8)	(8.4)	(17.0)	(79.7)
error<50ms				92.33%
error<10ms				56.64%

Table B.25.: Alignment Results k.330-3

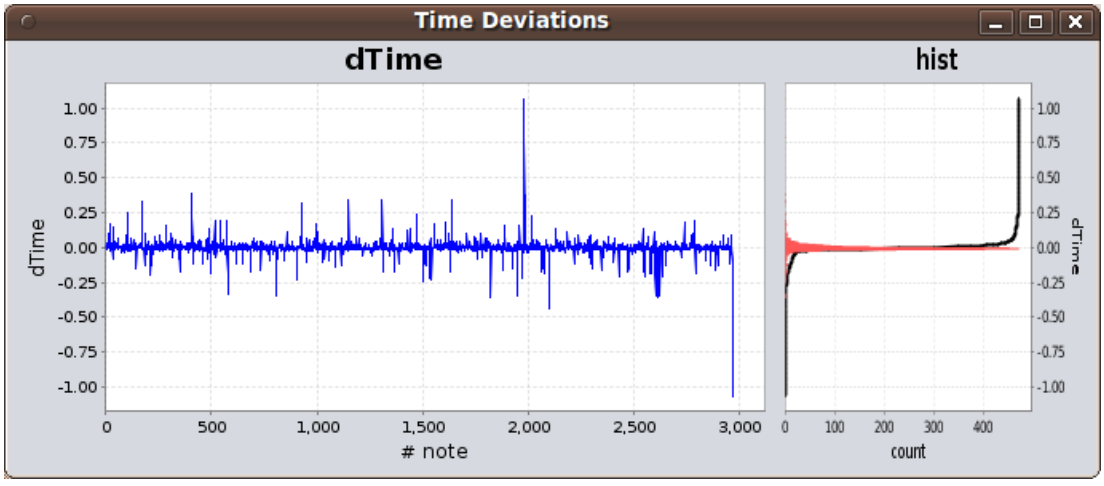


Figure B.21.: Time Deviations k.330-3

k.331 – 1^{st} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			6.2	(55.7)
std-dev			151.3	(140.8)
min-error			-1039.7	(0.0)
max-error			2619.7	(2619.7)
-99.6	-21.0	-6.1	9.2	163.5
(1.4)	(7.0)	(16.3)	(44.9)	(227.4)
error<50ms				77.46%
error<10ms				35.77%

(b) Refined Anchor Notes

avg-error			7.2	(25.4)
std-dev			107.2	(104.4)
min-error			-991.8	(0.0)
max-error			2433.8	(2433.8)
-12.4	-0.5	4.6	13.8	45.5
(0.5)	(2.7)	(6.7)	(16.2)	(63.3)
error<50ms				93.10%
error<10ms				61.14%

(c) Non-Anchor Notes before Refinement

avg-error			6.8	(66.3)
std-dev			166.5	(152.9)
min-error			-1039.7	(0.0)
max-error			2619.7	(2619.7)
-128.4	-26.9	-9.0	6.8	201.8
(1.8)	(8.5)	(20.6)	(56.0)	(268.6)
error<50ms				73.11%
error<10ms				29.62%

(d) Non-Anchor Notes after Refinement

avg-error			-0.6	(40.8)
std-dev			134.8	(128.5)
min-error			-1480.7	(0.0)
max-error			2321.9	(2321.9)
-97.3	-8.1	-1.6	8.2	79.5
(0.7)	(3.6)	(8.1)	(23.5)	(202.0)
error<50ms				85.72%
error<10ms				57.58%

(e) Complete System

avg-error			2.2	(36.4)
std-dev			127.0	(121.7)
min-error			-1480.7	(0.0)
max-error			2433.8	(2433.8)
-69.2	-5.8	1.0	11.4	62.0
(0.7)	(3.4)	(8.1)	(20.4)	(146.7)
error<50ms				87.85%
error<10ms				57.50%

Table B.26.: Alignment Results k.331-1

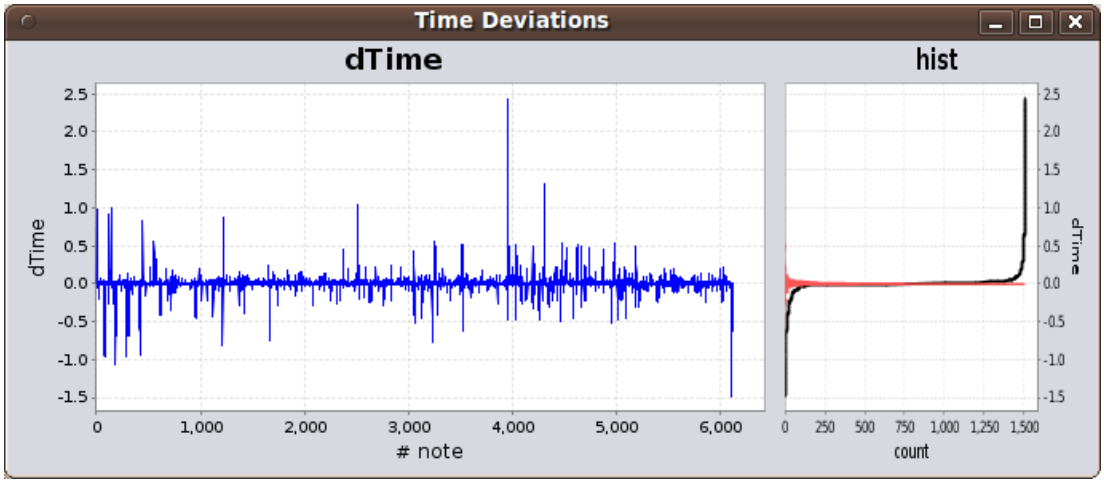


Figure B.22.: Time Deviations k.331-1

k.331 – 2^{nd} Movement

(a) Two-scale DTW and Chroma vectors

avg-error				2.9 (36.1)
std-dev				81.0 (72.6)
min-error				-461.8 (0.0)
max-error				787.2 (787.2)
-58.3 (1.0)	-15.0 (5.8)	-3.7 (13.6)	11.1 (31.5)	100.9 (160.4)
error<50ms				84.03%
error<10ms				40.63%

(b) Refined Anchor Notes

avg-error				6.9 (16.6)
std-dev				51.5 (49.2)
min-error				-449.6 (0.0)
max-error				829.0 (829.0)
-7.0 (0.5)	0.8 (2.8)	5.2 (6.3)	12.6 (14.7)	36.8 (48.3)
error<50ms				95.25%
error<10ms				64.96%

(c) Non-Anchor Notes before Refinement

avg-error				-1.4 (43.8)
std-dev				93.9 (83.1)
min-error				-461.8 (0.0)
max-error				787.2 (787.2)
-92.8 (1.6)	-21.3 (7.2)	-8.4 (16.3)	2.8 (36.7)	134.4 (204.5)
error<50ms				81.80%
error<10ms				34.43%

(d) Non-Anchor Notes after Refinement

avg-error				0.5 (27.0)
std-dev				85.3 (80.9)
min-error				-725.5 (0.0)
max-error				1578.7 (1578.7)
-50.0 (0.6)	-7.6 (3.6)	-1.4 (7.8)	7.9 (19.3)	54.9 (106.5)
error<50ms				89.54%
error<10ms				58.76%

(e) Complete System

avg-error				3.4 (23.3)
std-dev				74.0 (70.3)
min-error				-725.5 (0.0)
max-error				1578.7 (1578.7)
-31.5 (0.6)	-4.7 (3.3)	2.2 (7.5)	11.3 (17.9)	47.6 (89.0)
error<50ms				91.53%
error<10ms				59.90%

Table B.27.: Alignment Results k.331-2

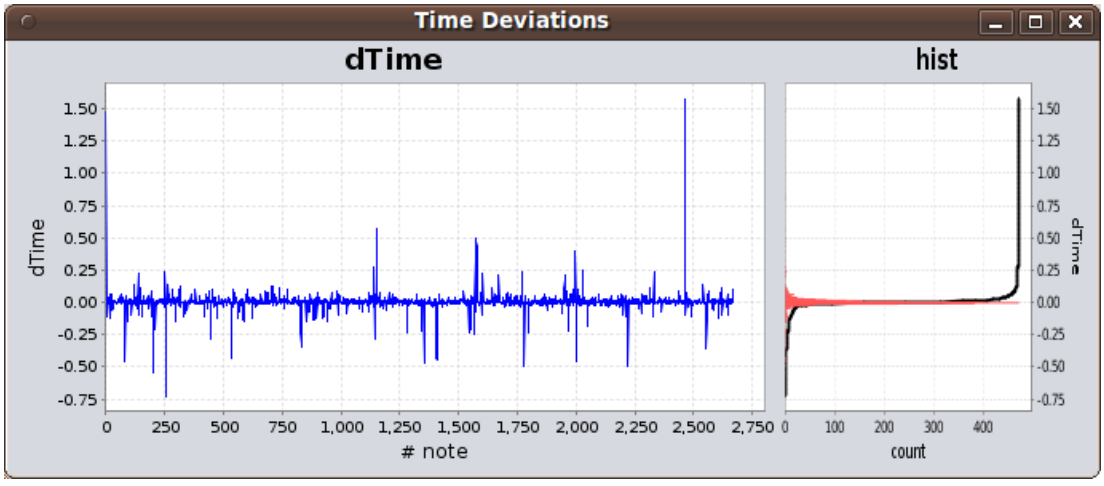


Figure B.23.: Time Deviations k.331-2

k.331 – 3rd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			7.1	(42.5)
std-dev			113.4	(105.4)
min-error			-368.7	(0.0)
max-error			1665.2	(1665.2)
-84.8 (1.5)	-14.7 (6.7)	-5.0 (14.3)	13.7 (41.6)	110.0 (171.7)
error<50ms				77.93%
error<10ms				37.21%

(b) Refined Anchor Notes

avg-error			7.1	(22.3)
std-dev			49.9	(45.2)
min-error			-391.9	(0.0)
max-error			341.6	(391.9)
-13.2 (0.6)	-1.5 (3.3)	5.2 (7.8)	17.0 (20.4)	51.6 (94.9)
error<50ms				90.56%
error<10ms				56.38%

(c) Non-Anchor Notes before Refinement

avg-error			7.4	(47.0)
std-dev			128.5	(119.8)
min-error			-340.9	(0.0)
max-error			1665.2	(1665.2)
-102.1 (1.7)	-16.7 (7.6)	-6.5 (15.5)	12.5 (43.3)	134.1 (184.7)
error<50ms				77.25%
error<10ms				33.94%

(d) Non-Anchor Notes after Refinement

avg-error			0.6	(38.8)
std-dev			125.9	(119.8)
min-error			-1012.1	(0.0)
max-error			1665.2	(1665.2)
-118.9 (0.7)	-9.0 (4.0)	-2.7 (8.7)	8.1 (29.6)	79.6 (164.8)
error<50ms				82.28%
error<10ms				54.74%

(e) Complete System

avg-error			3.1	(34.9)
std-dev			110.8	(105.2)
min-error			-1012.1	(0.0)
max-error			1665.2	(1665.2)
-104.7 (0.7)	-7.1 (3.9)	-0.3 (8.6)	12.1 (27.2)	81.9 (141.6)
error<50ms				83.57%
error<10ms				54.68%

Table B.28.: Alignment Results k.331-3

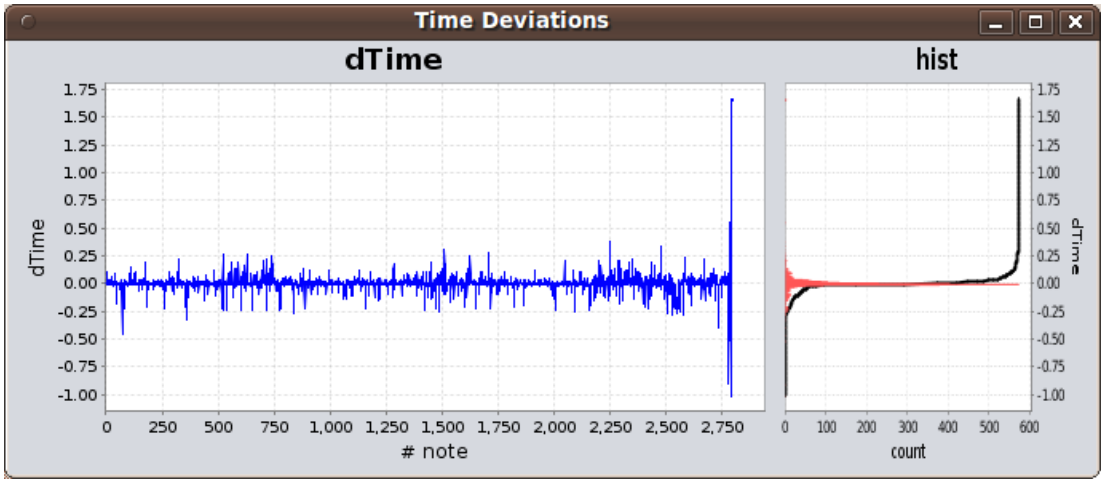


Figure B.24.: Time Deviations k.331-3

k.332 – 1^{st} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			17.1	(117.7)
std-dev			252.5	(224.0)
min-error			-1425.5	(0.0)
max-error			2115.7	(2115.7)
-280.1	-26.7	-8.4	31.3	380.1
(2.5)	(10.8)	(27.6)	(111.1)	(547.3)
error<50ms				61.56%
error<10ms				23.11%

(b) Refined Anchor Notes

avg-error			11.0	(40.4)
std-dev			133.4	(127.6)
min-error			-1218.6	(0.0)
max-error			1804.2	(1804.2)
-30.2	-2.6	2.9	15.6	76.3
(0.4)	(2.7)	(6.8)	(20.7)	(199.1)
error<50ms				88.53%
error<10ms				60.00%

(c) Non-Anchor Notes before Refinement

avg-error			19.0	(144.2)
std-dev			286.5	(248.3)
min-error			-1425.5	(0.0)
max-error			2115.7	(2115.7)
-346.9	-34.8	-11.0	33.8	437.9
(3.1)	(13.0)	(34.7)	(163.5)	(674.8)
error<50ms				55.81%
error<10ms				19.03%

(d) Non-Anchor Notes after Refinement

avg-error			7.2	(84.0)
std-dev			216.7	(199.9)
min-error			-1125.0	(0.0)
max-error			1962.8	(1962.8)
-224.4	-13.7	-3.4	13.4	277.6
(1.0)	(6.2)	(13.7)	(46.3)	(412.1)
error<50ms				75.77%
error<10ms				39.92%

(e) Complete System

avg-error			8.5	(71.8)
std-dev			196.6	(183.2)
min-error			-1218.6	(0.0)
max-error			1962.8	(1962.8)
-192.4	-10.3	-0.4	15.0	205.2
(0.8)	(4.8)	(11.8)	(36.5)	(383.3)
error<50ms				79.31%
error<10ms				44.64%

Table B.29.: Alignment Results k.332-1

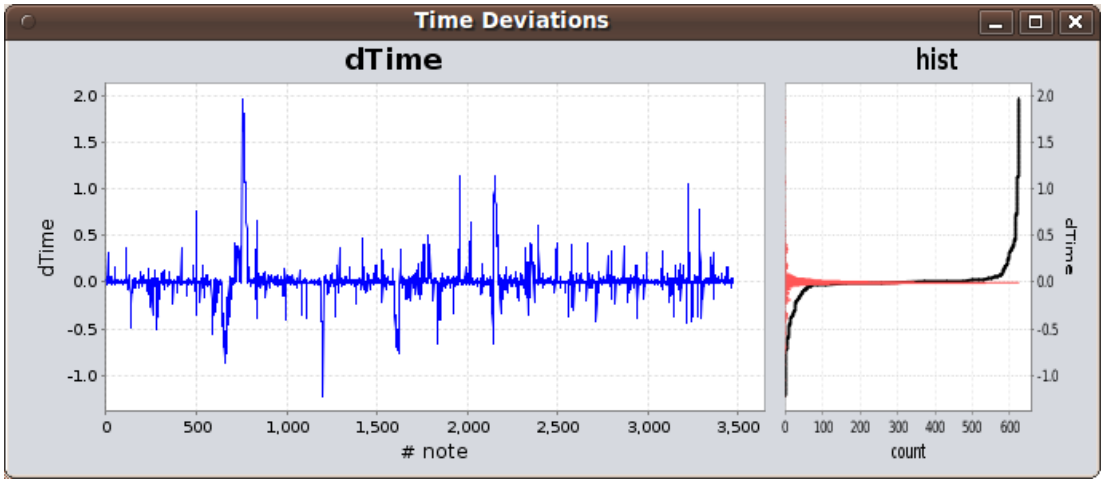


Figure B.25.: Time Deviations k.332-1

k.332 – 2^{nd} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			-29.2	(142.8)
std-dev			309.9	(276.5)
min-error			-1759.8	(0.0)
max-error			2528.1	(2528.1)
-488.1 (2.3)	-49.8 (12.7)	-12.4 (35.6)	13.1 (130.4)	335.8 (621.0)
error<50ms				58.97%
error<10ms				19.58%

(b) Refined Anchor Notes

avg-error			17.1	(59.6)
std-dev			245.6	(238.9)
min-error			-1734.7	(0.0)
max-error			2539.7	(2539.7)
-15.7 (0.5)	-0.8 (2.4)	4.3 (5.8)	12.3 (17.2)	58.2 (187.0)
error<50ms				90.20%
error<10ms				65.20%

(c) Non-Anchor Notes before Refinement

avg-error			-49.8	(175.5)
std-dev			333.1	(287.5)
min-error			-1759.8	(0.1)
max-error			2143.3	(2143.3)
-530.2 (4.4)	-99.6 (17.9)	-25.0 (49.1)	-1.0 (212.8)	385.2 (721.7)
error<50ms				50.98%
error<10ms				13.46%

(d) Non-Anchor Notes after Refinement

avg-error			-10.3	(103.4)
std-dev			300.1	(281.9)
min-error			-2868.2	(0.1)
max-error			1724.6	(2868.2)
-407.7 (1.2)	-11.3 (4.9)	-3.9 (10.8)	8.1 (38.0)	251.9 (620.6)
error<50ms				79.40%
error<10ms				47.99%

(e) Complete System

avg-error			-1.1	(89.8)
std-dev			284.2	(269.6)
min-error			-2868.2	(0.0)
max-error			2539.7	(2868.2)
-264.9 (0.8)	-8.7 (4.0)	-0.4 (9.5)	11.5 (28.3)	182.9 (540.9)
error<50ms				82.77%
error<10ms				52.31%

Table B.30.: Alignment Results k.332-2

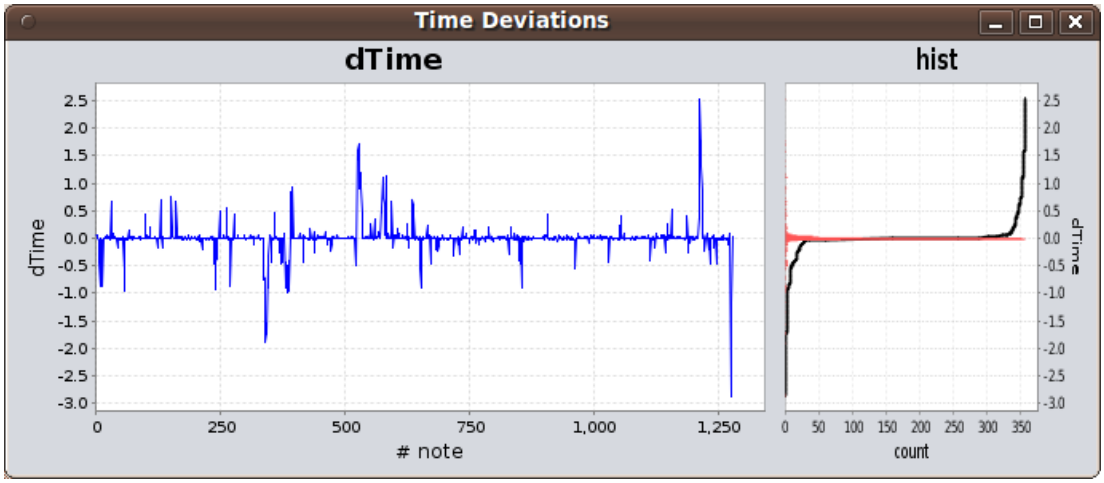


Figure B.26.: Time Deviations k.332-2

k.332 – 3rd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			15.0	(62.8)
std-dev			147.0	(133.7)
min-error			-797.8	(0.0)
max-error			1385.0	(1385.0)
-125.2 (1.7)	-15.4 (7.9)	-5.6 (16.9)	20.4 (52.6)	204.6 (281.8)
error<50ms				74.08%
error<10ms				32.32%

(b) Refined Anchor Notes

avg-error			9.2	(20.8)
std-dev			73.6	(71.2)
min-error			-407.6	(0.0)
max-error			1275.6	(1275.6)
-8.8 (0.4)	-2.2 (2.7)	3.4 (6.1)	11.9 (14.3)	40.5 (68.1)
error<50ms				93.33%
error<10ms				66.67%

(c) Non-Anchor Notes before Refinement

avg-error			16.7	(78.7)
std-dev			171.0	(152.8)
min-error			-797.8	(0.0)
max-error			1385.0	(1385.0)
-173.0 (2.7)	-20.2 (9.5)	-8.1 (20.0)	19.3 (69.6)	248.4 (361.3)
error<50ms				69.05%
error<10ms				26.52%

(d) Non-Anchor Notes after Refinement

avg-error			6.4	(42.0)
std-dev			127.0	(120.0)
min-error			-1883.4	(0.0)
max-error			1703.6	(1883.4)
-88.8 (1.1)	-11.3 (5.3)	-3.7 (11.1)	10.7 (29.2)	94.1 (193.9)
error<50ms				83.43%
error<10ms				45.98%

(e) Complete System

avg-error			7.4	(35.0)
std-dev			111.9	(106.5)
min-error			-1883.4	(0.0)
max-error			1703.6	(1883.4)
-60.4 (0.8)	-8.0 (4.2)	-0.4 (9.3)	11.6 (22.4)	74.2 (170.5)
error<50ms				86.72%
error<10ms				52.31%

Table B.31.: Alignment Results k.332-3

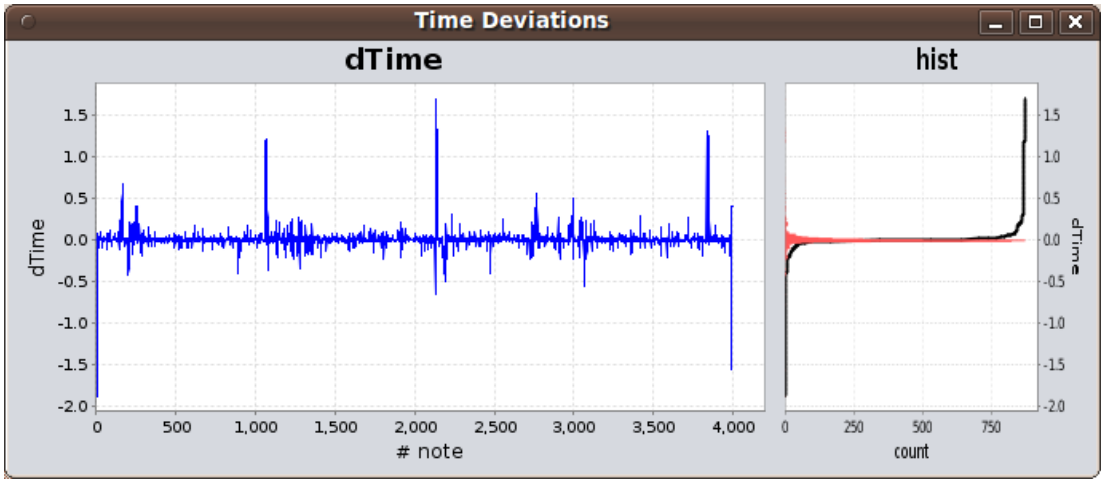


Figure B.27.: Time Deviations k.332-3

k.333 – 1^{st} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			14.3	(43.5)
std-dev			116.7	(109.2)
min-error			-886.5	(0.0)
max-error			1577.1	(1577.1)
-67.0	-12.1	-2.9	16.5	131.6
(1.0)	(6.2)	(13.4)	(37.4)	(179.4)
error<50ms				80.92%
error<10ms				40.49%

(b) Refined Anchor Notes

avg-error			8.9	(15.6)
std-dev			38.3	(36.1)
min-error			-299.0	(0.0)
max-error			469.1	(469.1)
-5.3	0.2	5.3	13.3	36.0
(0.6)	(2.5)	(6.1)	(14.4)	(46.5)
error<50ms				95.25%
error<10ms				65.27%

(c) Non-Anchor Notes before Refinement

avg-error			16.2	(53.3)
std-dev			139.8	(130.2)
min-error			-886.5	(0.0)
max-error			1577.1	(1577.1)
-91.4	-15.3	-6.2	11.8	186.2
(1.6)	(7.2)	(14.6)	(41.1)	(216.4)
error<50ms				78.37%
error<10ms				36.45%

(d) Non-Anchor Notes after Refinement

avg-error			2.3	(25.9)
std-dev			60.1	(54.2)
min-error			-520.1	(0.0)
max-error			541.8	(541.8)
-54.0	-7.9	-1.3	9.4	61.5
(0.8)	(3.7)	(8.4)	(21.0)	(111.7)
error<50ms				88.05%
error<10ms				57.37%

(e) Complete System

avg-error			4.5	(22.5)
std-dev			53.8	(49.1)
min-error			-520.1	(0.0)
max-error			541.8	(541.8)
-29.4	-5.2	1.4	11.6	53.7
(0.7)	(3.3)	(7.6)	(17.8)	(99.1)
error<50ms				90.46%
error<10ms				59.91%

Table B.32.: Alignment Results k.333-1

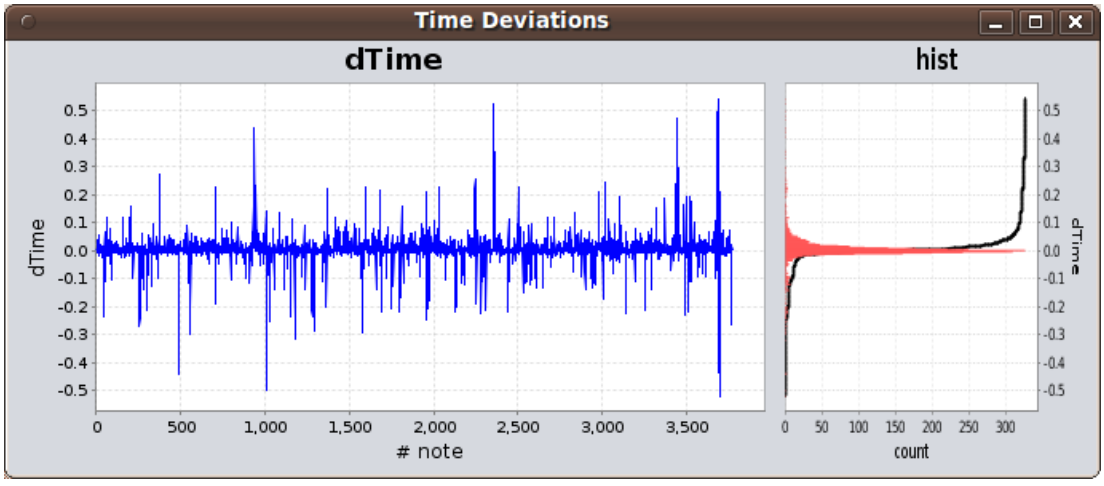


Figure B.28.: Time Deviations k.333-1

k.333 – 2^{nd} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			-7.0	(130.4)
std-dev			265.2	(231.0)
min-error			-1474.1	(0.0)
max-error			1767.6	(1767.6)
-426.6 (3.0)	-40.6 (12.4)	-9.9 (34.0)	22.1 (115.9)	423.6 (625.7)
error<50ms				60.90%
error<10ms				19.47%

(b) Refined Anchor Notes

avg-error			11.1	(40.6)
std-dev			151.4	(146.3)
min-error			-991.2	(0.0)
max-error			1282.8	(1282.8)
-15.1 (0.4)	-0.7 (2.2)	4.3 (6.4)	14.8 (17.5)	57.0 (122.7)
error<50ms				91.47%
error<10ms				61.73%

(c) Non-Anchor Notes before Refinement

avg-error			-18.0	(177.2)
std-dev			314.2	(260.1)
min-error			-1474.1	(0.1)
max-error			1767.6	(1767.6)
-569.5 (5.9)	-71.5 (18.2)	-19.6 (51.0)	11.5 (248.3)	559.3 (694.1)
error<50ms				49.55%
error<10ms				10.11%

(d) Non-Anchor Notes after Refinement

avg-error			-4.2	(93.5)
std-dev			234.2	(214.8)
min-error			-1447.6	(0.1)
max-error			1773.4	(1773.4)
-349.2 (1.5)	-13.0 (5.9)	-4.6 (12.4)	10.2 (41.1)	283.9 (649.8)
error<50ms				77.51%
error<10ms				42.95%

(e) Complete System

avg-error			2.3	(74.0)
std-dev			206.9	(193.3)
min-error			-1447.6	(0.0)
max-error			1773.4	(1773.4)
-201.2 (0.9)	-8.3 (4.4)	0.6 (10.9)	15.1 (28.8)	150.9 (607.2)
error<50ms				82.50%
error<10ms				48.20%

Table B.33.: Alignment Results k.333-2

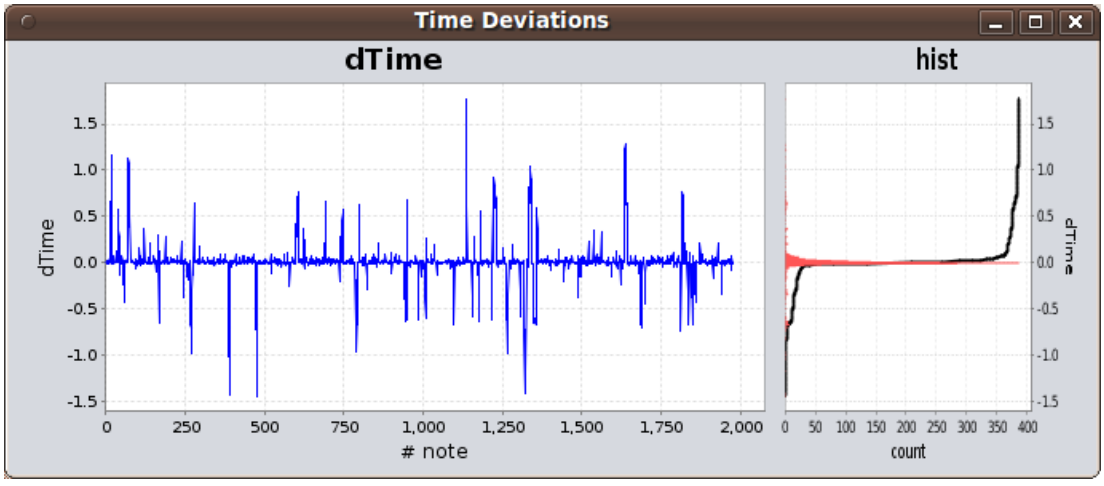


Figure B.29.: Time Deviations k.333-2

k.333 – 3rd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			10.9	(77.3)
std-dev			204.2	(189.3)
min-error			-1297.3	(0.0)
max-error			2343.1	(2343.1)
-136.0 (1.9)	-17.7 (8.2)	-5.5 (17.7)	17.6 (59.6)	234.5 (358.9)
error<50ms				72.54%
error<10ms				31.57%

(b) Refined Anchor Notes

avg-error			6.2	(30.1)
std-dev			100.6	(96.2)
min-error			-1250.3	(0.0)
max-error			858.0	(1250.3)
-5.6 (0.5)	0.8 (2.8)	5.9 (7.2)	15.2 (17.5)	45.2 (120.5)
error<50ms				92.41%
error<10ms				58.63%

(c) Non-Anchor Notes before Refinement

avg-error			13.7	(95.0)
std-dev			236.6	(217.1)
min-error			-1297.3	(0.0)
max-error			2343.1	(2343.1)
-187.8 (2.5)	-22.9 (9.5)	-8.4 (21.3)	16.9 (77.2)	283.3 (406.2)
error<50ms				67.93%
error<10ms				26.81%

(d) Non-Anchor Notes after Refinement

avg-error			3.7	(63.6)
std-dev			168.5	(156.0)
min-error			-1399.4	(0.0)
max-error			1948.4	(1948.4)
-167.9 (0.8)	-8.6 (4.5)	-2.1 (9.2)	11.5 (40.4)	191.4 (380.2)
error<50ms				77.98%
error<10ms				52.91%

(e) Complete System

avg-error			4.6	(53.4)
std-dev			150.8	(141.1)
min-error			-1399.4	(0.0)
max-error			1948.4	(1948.4)
-117.0 (0.7)	-6.1 (3.9)	1.4 (8.9)	14.3 (28.1)	125.2 (350.5)
error<50ms				82.44%
error<10ms				53.82%

Table B.34.: Alignment Results k.333-3

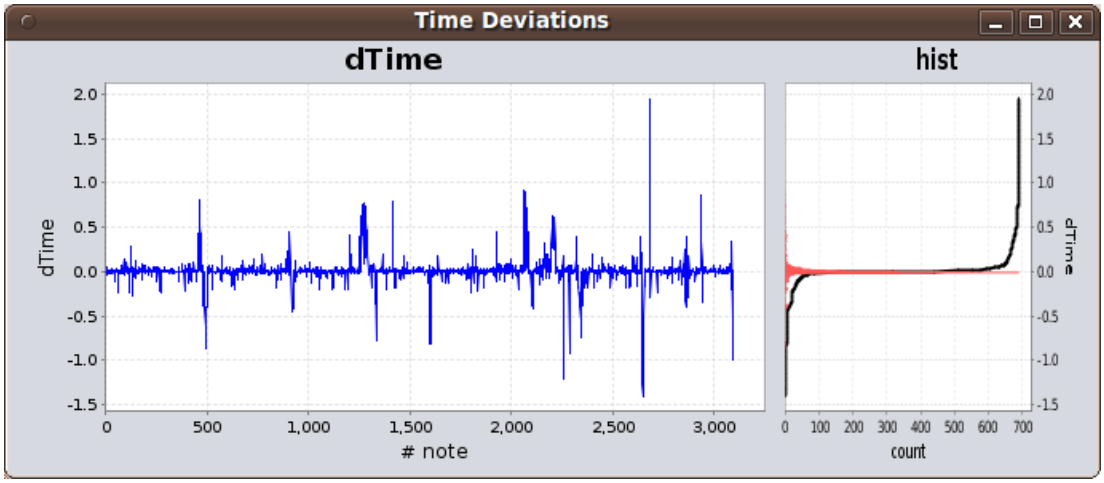


Figure B.30.: Time Deviations k.333-3

k.457 – 1st Movement

(a) Two-scale DTW and Chroma vectors

avg-error			29.9	(73.0)
std-dev			154.8	(139.8)
min-error			-973.4	(0.0)
max-error			1690.3	(1690.3)
-107.4 (1.6)	-14.6 (7.8)	-2.7 (18.5)	34.7 (72.6)	314.7 (345.3)
error<50ms				68.93%
error<10ms				32.31%

(b) Refined Anchor Notes

avg-error			15.4	(25.6)
std-dev			92.2	(89.9)
min-error			-520.9	(0.0)
max-error			1571.5	(1571.5)
-8.0 (0.5)	-1.0 (2.6)	4.3 (6.1)	14.8 (17.5)	54.2 (94.8)
error<50ms				92.16%
error<10ms				62.65%

(c) Non-Anchor Notes before Refinement

avg-error			36.9	(92.7)
std-dev			177.7	(156.1)
min-error			-973.4	(0.0)
max-error			1690.3	(1690.3)
-153.5 (2.4)	-18.6 (9.8)	-5.4 (24.7)	53.4 (114.2)	353.6 (377.0)
error<50ms				61.89%
error<10ms				25.85%

(d) Non-Anchor Notes after Refinement

avg-error			11.2	(58.4)
std-dev			180.1	(170.8)
min-error			-1756.2	(0.0)
max-error			2723.5	(2723.5)
-111.5 (1.2)	-12.6 (6.2)	-4.5 (13.2)	14.0 (40.0)	156.4 (269.7)
error<50ms				79.07%
error<10ms				40.34%

(e) Complete System

avg-error			12.8	(47.3)
std-dev			155.9	(149.1)
min-error			-1756.2	(0.0)
max-error			2723.5	(2723.5)
-81.4 (0.8)	-8.8 (4.7)	0.3 (10.9)	14.7 (28.7)	110.3 (190.2)
error<50ms				83.53%
error<10ms				47.41%

Table B.35.: Alignment Results k.457-1

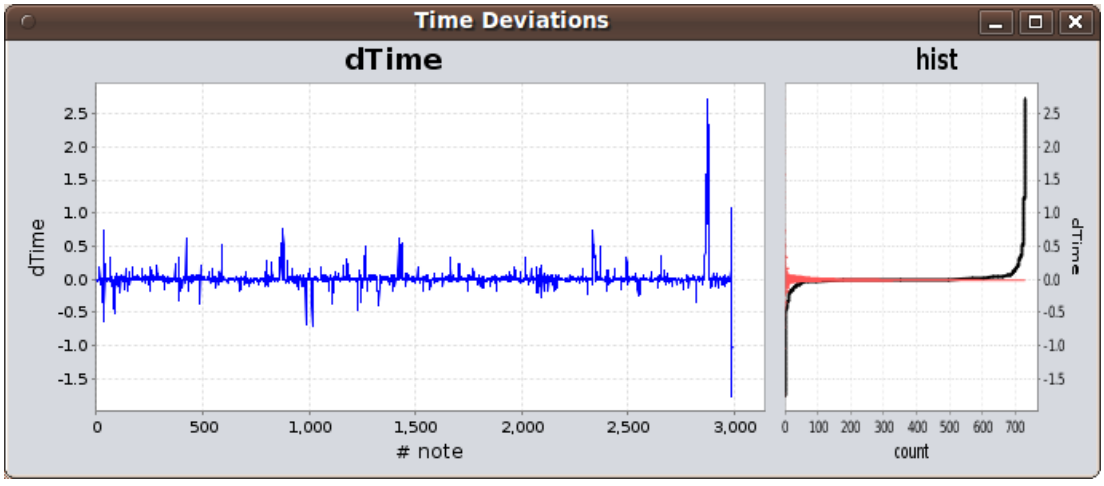


Figure B.31.: Time Deviations k.457-1

k.457 – 2nd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			-80.6	(182.2)
std-dev			393.1	(357.5)
min-error			-3386.5	(0.0)
max-error			2327.8	(3386.5)
-784.0 (3.5)	-74.3 (16.3)	-18.9 (38.7)	6.1 (181.1)	306.6 (912.2)
error<50ms				55.77%
error<10ms				12.91%

(b) Refined Anchor Notes

avg-error			-27.0	(57.6)
std-dev			242.1	(236.7)
min-error			-3045.1	(0.0)
max-error			1269.9	(3045.1)
-233.5 (0.6)	-6.4 (3.3)	-1.3 (6.6)	7.1 (12.1)	37.4 (312.1)
error<50ms				91.09%
error<10ms				67.90%

(c) Non-Anchor Notes before Refinement

avg-error			-102.4	(232.2)
std-dev			442.6	(390.4)
min-error			-3386.5	(0.0)
max-error			2327.8	(3386.5)
-888.8 (8.3)	-142.9 (22.2)	-27.4 (68.1)	-3.5 (271.0)	394.6 (944.7)
error<50ms				44.67%
error<10ms				6.44%

(d) Non-Anchor Notes after Refinement

avg-error			-76.7	(161.9)
std-dev			414.1	(388.8)
min-error			-3118.2	(0.1)
max-error			1817.0	(3118.2)
-895.0 (2.4)	-20.1 (9.9)	-11.5 (16.3)	2.0 (49.4)	242.0 (961.7)
error<50ms				75.14%
error<10ms				25.43%

(e) Complete System

avg-error			-61.2	(129.9)
std-dev			370.5	(352.3)
min-error			-3118.2	(0.0)
max-error			1817.0	(3118.2)
-631.0 (1.3)	-16.3 (6.5)	-6.9 (13.4)	5.5 (32.0)	159.4 (901.9)
error<50ms				80.00%
error<10ms				38.29%

Table B.36.: Alignment Results k.457-2

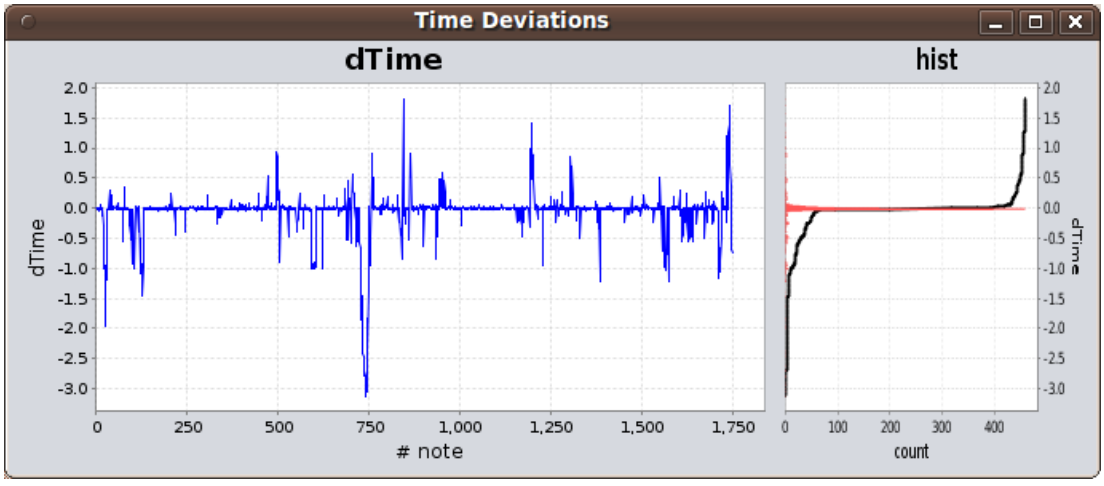


Figure B.32.: Time Deviations k.457-2

k.457 – 3rd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			15.2	(93.5)
std-dev			192.5	(168.9)
min-error			-566.9	(0.0)
max-error			1780.2	(1780.2)
-208.4 (3.1)	-33.7 (13.6)	-11.3 (33.2)	32.5 (106.1)	271.1 (344.3)
error<50ms				58.75%
error<10ms				16.48%

(b) Refined Anchor Notes

avg-error			2.1	(29.1)
std-dev			94.5	(89.9)
min-error			-539.2	(0.0)
max-error			1660.6	(1660.6)
-17.4 (0.9)	-8.1 (4.0)	-3.1 (7.7)	5.6 (13.2)	52.3 (234.3)
error<50ms				90.78%
error<10ms				63.26%

(c) Non-Anchor Notes before Refinement

avg-error			20.8	(117.8)
std-dev			224.4	(192.1)
min-error			-566.9	(0.0)
max-error			1780.2	(1780.2)
-240.1 (4.5)	-51.2 (17.8)	-14.5 (50.1)	47.5 (153.5)	322.4 (429.3)
error<50ms				49.97%
error<10ms				12.00%

(d) Non-Anchor Notes after Refinement

avg-error			10.2	(71.9)
std-dev			200.3	(187.2)
min-error			-982.8	(0.0)
max-error			2356.8	(2356.8)
-193.1 (2.6)	-21.9 (10.6)	-11.2 (19.3)	10.0 (43.0)	235.6 (285.5)
error<50ms				77.37%
error<10ms				22.36%

(e) Complete System

avg-error			7.9	(58.1)
std-dev			173.3	(163.4)
min-error			-982.8	(0.0)
max-error			2356.8	(2356.8)
-133.9 (1.6)	-17.1 (7.3)	-6.9 (14.8)	8.8 (32.0)	168.9 (268.4)
error<50ms				81.72%
error<10ms				35.08%

Table B.37.: Alignment Results k.457-3

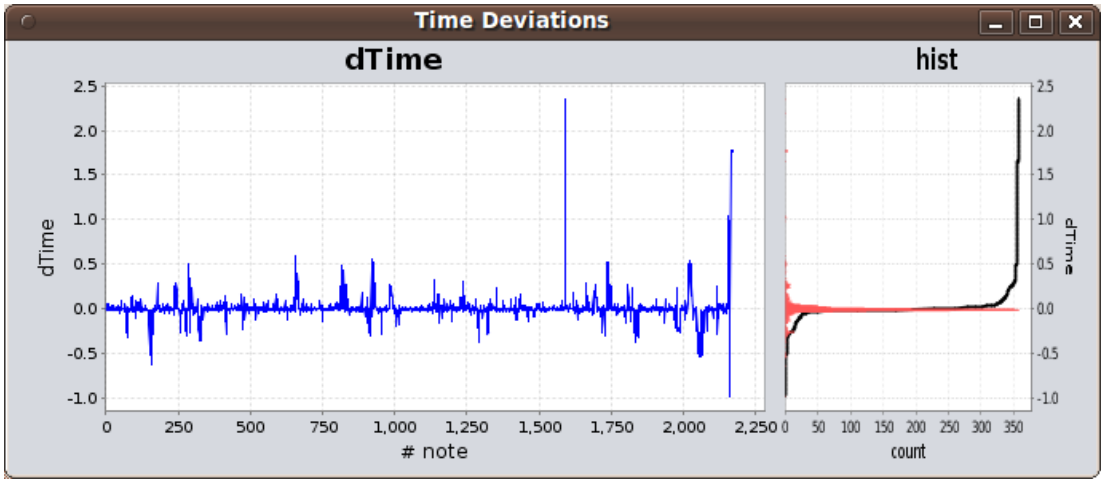


Figure B.33.: Time Deviations k.457-3

k.475 – 1^{st} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			25.5	(317.7)
std-dev			575.5	(480.6)
min-error			-2216.4	(0.0)
max-error			2943.7	(2943.7)
-802.2	-121.6	-15.8	55.8	1232.9
(4.8)	(19.0)	(94.2)	(393.0)	(1365.3)
error<50ms				41.73%
error<10ms				12.44%

(b) Refined Anchor Notes

avg-error			2.8	(152.0)
std-dev			392.9	(362.3)
min-error			-1816.3	(0.1)
max-error			2035.4	(2035.4)
-558.9	-5.3	1.1	10.1	366.3
(0.6)	(3.0)	(7.3)	(34.6)	(949.3)
error<50ms				75.47%
error<10ms				56.98%

(c) Non-Anchor Notes before Refinement

avg-error			31.9	(358.3)
std-dev			613.8	(499.4)
min-error			-2216.4	(0.0)
max-error			2943.7	(2943.7)
-838.6	-165.7	-21.2	123.3	1252.7
(7.3)	(28.4)	(145.2)	(488.6)	(1536.9)
error<50ms				34.89%
error<10ms				7.87%

(d) Non-Anchor Notes after Refinement

avg-error			-16.7	(281.7)
std-dev			547.3	(469.6)
min-error			-2179.9	(0.0)
max-error			2907.6	(2907.6)
-823.1	-23.7	-5.6	20.5	1201.6
(1.9)	(7.4)	(22.9)	(393.3)	(1303.6)
error<50ms				60.93%
error<10ms				33.82%

(e) Complete System

avg-error			-12.6	(255.2)
std-dev			519.3	(452.5)
min-error			-2179.9	(0.0)
max-error			2907.6	(2907.6)
-792.1	-17.0	-3.5	16.7	1176.9
(1.5)	(6.2)	(16.9)	(363.6)	(1273.9)
error<50ms				63.83%
error<10ms				38.18%

Table B.38.: Alignment Results k.475-1

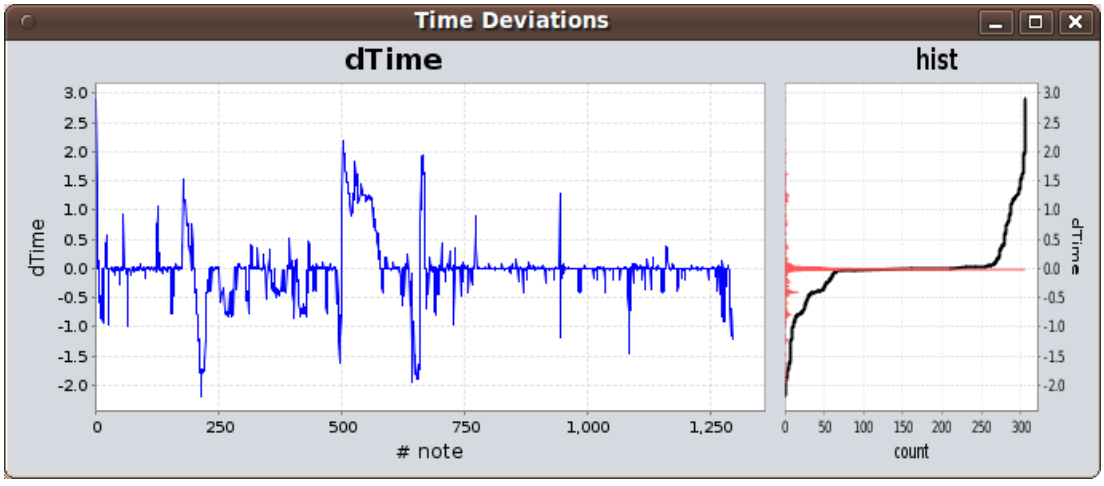


Figure B.34.: Time Deviations k.475-1

k.475 – 2nd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			18.4	(119.0)
std-dev			295.9	(271.6)
min-error			-2615.1	(0.0)
max-error			2132.1	(2615.1)
-261.1 (1.5)	-19.4 (9.8)	0.2 (27.2)	43.3 (101.7)	344.2 (522.5)
error<50ms				63.01%
error<10ms				25.74%

(b) Refined Anchor Notes

avg-error			17.2	(54.8)
std-dev			222.5	(216.4)
min-error			-1555.7	(0.0)
max-error			2180.2	(2180.2)
-13.8 (0.7)	-4.8 (3.5)	0.8 (7.4)	11.3 (16.8)	102.7 (208.4)
error<50ms				87.58%
error<10ms				62.75%

(c) Non-Anchor Notes before Refinement

avg-error			13.9	(144.1)
std-dev			324.2	(290.7)
min-error			-2615.1	(0.0)
max-error			2132.1	(2615.1)
-331.1 (2.0)	-30.5 (12.7)	-3.9 (34.7)	46.3 (148.6)	440.7 (644.6)
error<50ms				56.60%
error<10ms				20.17%

(d) Non-Anchor Notes after Refinement

avg-error			26.3	(93.0)
std-dev			279.4	(264.7)
min-error			-1564.4	(0.1)
max-error			2161.1	(2161.1)
-164.6 (1.7)	-13.0 (7.1)	-3.9 (14.9)	21.0 (41.9)	225.3 (503.2)
error<50ms				76.34%
error<10ms				34.57%

(e) Complete System

avg-error			23.5	(80.6)
std-dev			261.6	(250.0)
min-error			-1564.4	(0.0)
max-error			2161.1	(2161.1)
-108.8 (1.1)	-10.6 (5.3)	-1.4 (11.7)	17.0 (34.3)	190.2 (380.2)
error<50ms				80.07%
error<10ms				43.24%

Table B.39.: Alignment Results k.475-2

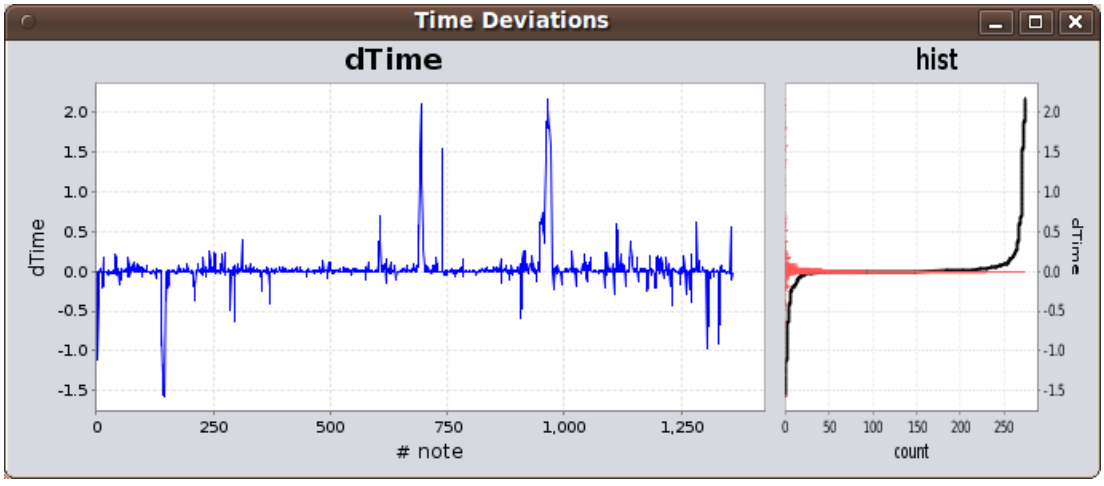


Figure B.35.: Time Deviations k.475-2

k.475 – 3rd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			12.3	(252.6)
std-dev			482.7	(411.5)
min-error			-2473.2	(0.1)
max-error			4479.9	(4479.9)
-697.7 (5.5)	-72.4 (24.1)	-14.0 (76.6)	88.6 (349.4)	728.0 (959.6)
error<50ms				41.67%
error<10ms				10.69%

(b) Refined Anchor Notes

avg-error			25.2	(125.7)
std-dev			323.8	(299.4)
min-error			-2386.1	(0.0)
max-error			1851.3	(2386.1)
-401.8 (0.6)	-2.9 (4.1)	5.2 (11.9)	24.3 (49.4)	473.8 (697.7)
error<50ms				75.40%
error<10ms				45.56%

(c) Non-Anchor Notes before Refinement

avg-error			10.1	(280.2)
std-dev			513.4	(430.2)
min-error			-2471.1	(0.1)
max-error			4479.9	(4479.9)
-785.2 (7.0)	-92.8 (28.4)	-17.4 (110.7)	128.4 (383.4)	752.8 (1004.4)
error<50ms				36.62%
error<10ms				8.20%

(d) Non-Anchor Notes after Refinement

avg-error			49.8	(200.5)
std-dev			434.5	(388.7)
min-error			-2438.9	(0.1)
max-error			3776.3	(3776.3)
-523.7 (2.2)	-15.0 (8.7)	-2.9 (25.8)	48.0 (234.4)	776.1 (887.8)
error<50ms				60.74%
error<10ms				28.12%

(e) Complete System

avg-error			45.2	(186.0)
std-dev			415.5	(374.2)
min-error			-2438.9	(0.0)
max-error			3776.3	(3776.3)
-496.6 (1.6)	-13.0 (7.8)	0.1 (21.4)	39.8 (199.4)	699.9 (868.3)
error<50ms				63.60%
error<10ms				31.21%

Table B.40.: Alignment Results k.475-3

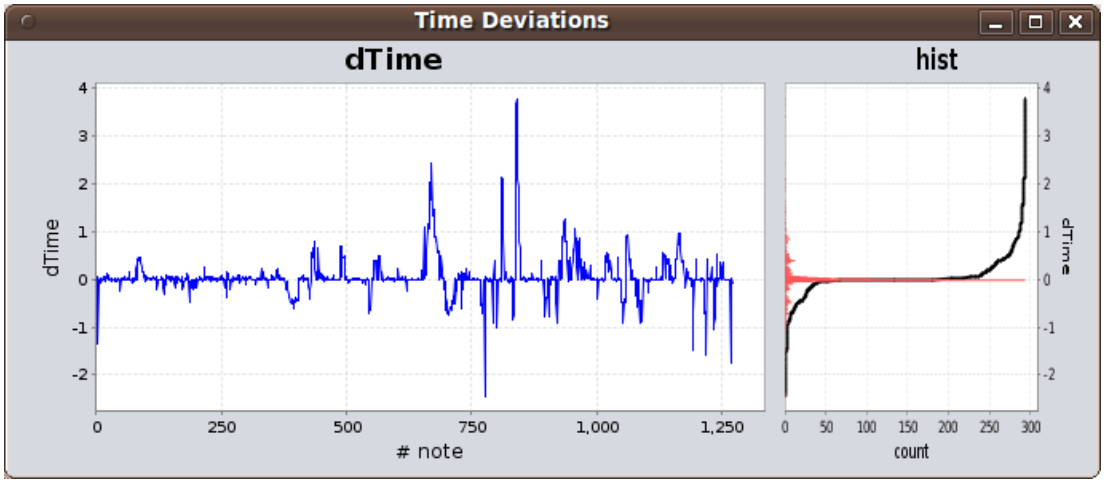


Figure B.36.: Time Deviations k.475-3

k.533 – 1^{st} Movement

(a) Two-scale DTW and Chroma vectors

avg-error			15.1	(59.7)
std-dev			144.0	(131.9)
min-error			-511.6	(0.0)
max-error			1824.7	(1824.7)
-109.4	-17.1	-6.1	18.0	188.5
(1.8)	(8.5)	(17.4)	(47.5)	(282.4)
error<50ms				75.92%
error<10ms				30.24%

(b) Refined Anchor Notes

avg-error			8.2	(17.9)
std-dev			63.2	(61.2)
min-error			-402.6	(0.0)
max-error			919.2	(919.2)
-7.9	-1.6	3.3	10.7	42.2
(0.4)	(2.4)	(5.3)	(12.0)	(55.3)
error<50ms				94.51%
error<10ms				70.53%

(c) Non-Anchor Notes before Refinement

avg-error			18.0	(75.6)
std-dev			170.6	(153.9)
min-error			-511.6	(0.0)
max-error			1824.7	(1824.7)
-155.6	-22.9	-9.4	13.8	275.1
(2.4)	(9.9)	(20.5)	(60.3)	(363.7)
error<50ms				70.97%
error<10ms				25.25%

(d) Non-Anchor Notes after Refinement

avg-error			13.7	(46.9)
std-dev			134.0	(126.2)
min-error			-1788.9	(0.0)
max-error			1047.2	(1788.9)
-77.3	-9.5	-2.7	10.1	116.4
(1.0)	(4.7)	(9.6)	(26.6)	(232.0)
error<50ms				84.81%
error<10ms				51.73%

(e) Complete System

avg-error			12.0	(37.1)
std-dev			114.9	(109.4)
min-error			-1788.9	(0.0)
max-error			1047.2	(1788.9)
-40.0	-6.8	0.2	11.0	74.4
(0.7)	(3.7)	(8.2)	(20.7)	(171.9)
error<50ms				88.11%
error<10ms				57.57%

Table B.41.: Alignment Results k.533-1

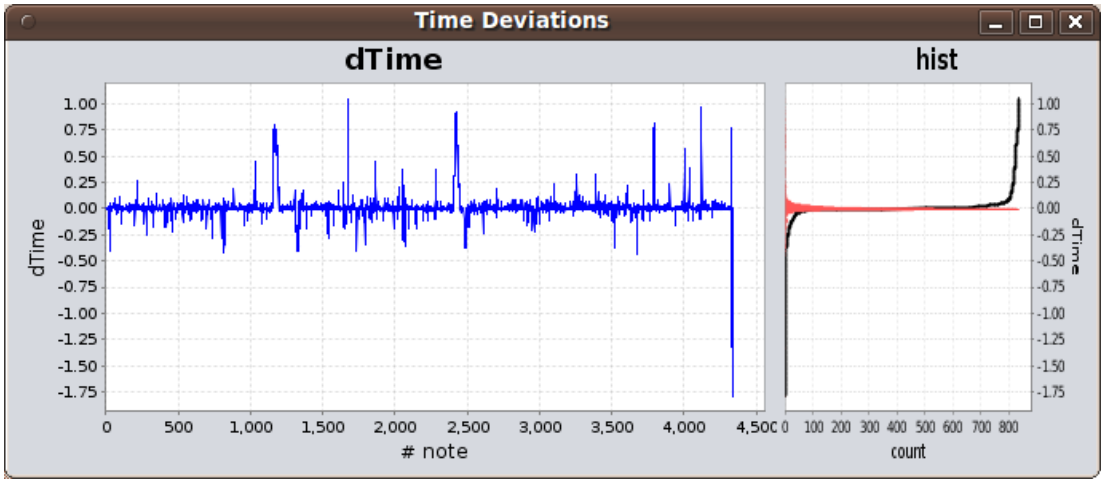


Figure B.37.: Time Deviations k.533-1

k.533 – 2nd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			-5.9	(87.5)
std-dev			202.9	(183.2)
min-error			-1455.5	(0.0)
max-error			1995.9	(1995.9)
-282.6 (2.0)	-23.8 (8.8)	-5.7 (22.6)	19.9 (67.0)	206.1 (411.1)
error<50ms				69.69%
error<10ms				28.69%

(b) Refined Anchor Notes

avg-error			-0.9	(26.7)
std-dev			112.0	(108.8)
min-error			-1513.6	(0.0)
max-error			554.6	(1513.6)
-9.2 (0.6)	-0.7 (2.7)	4.1 (5.9)	12.6 (15.0)	41.9 (54.0)
error<50ms				94.50%
error<10ms				66.11%

(c) Non-Anchor Notes before Refinement

avg-error			-12.5	(122.2)
std-dev			247.1	(215.1)
min-error			-1243.8	(0.0)
max-error			1995.9	(1995.9)
-371.0 (3.7)	-44.1 (12.1)	-12.8 (30.5)	7.8 (145.1)	320.8 (539.9)
error<50ms				61.16%
error<10ms				19.58%

(d) Non-Anchor Notes after Refinement

avg-error			-7.8	(50.7)
std-dev			155.3	(147.0)
min-error			-1614.2	(0.0)
max-error			1183.0	(1614.2)
-159.7 (1.0)	-10.0 (4.6)	-2.7 (9.6)	8.9 (27.0)	91.7 (219.4)
error<50ms				82.86%
error<10ms				51.01%

(e) Complete System

avg-error			-4.7	(41.3)
std-dev			139.3	(133.2)
min-error			-1614.2	(0.0)
max-error			1183.0	(1614.2)
-55.9 (0.8)	-6.4 (3.7)	1.2 (8.4)	12.5 (21.4)	69.2 (186.6)
error<50ms				87.44%
error<10ms				55.50%

Table B.42.: Alignment Results k.533-2

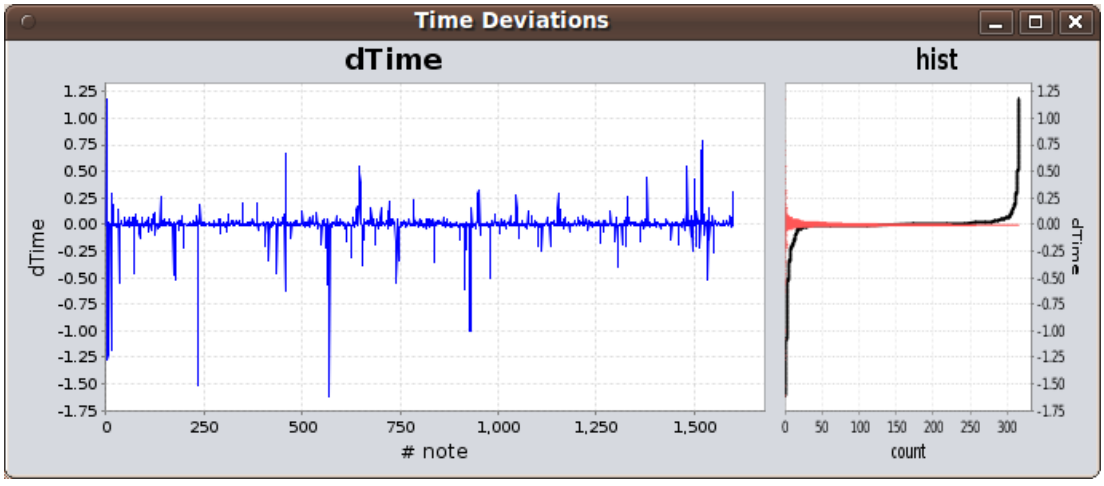


Figure B.38.: Time Deviations k.533-2

k.533 – 3rd Movement

(a) Two-scale DTW and Chroma vectors

avg-error			18.0	(66.1)
std-dev			163.5	(150.6)
min-error			-614.4	(0.0)
max-error			1747.6	(1747.6)
-106.3 (1.8)	-21.9 (8.7)	-7.9 (18.9)	12.1 (52.0)	235.2 (268.9)
error<50ms				74.29%
error<10ms				28.93%

(b) Refined Anchor Notes

avg-error			12.3	(24.0)
std-dev			101.3	(99.2)
min-error			-539.0	(0.0)
max-error			1414.5	(1414.5)
-8.0 (0.4)	-0.9 (2.1)	3.4 (5.1)	9.8 (11.6)	41.1 (84.3)
error<50ms				93.07%
error<10ms				70.84%

(c) Non-Anchor Notes before Refinement

avg-error			21.1	(81.5)
std-dev			185.7	(168.3)
min-error			-603.4	(0.1)
max-error			1747.6	(1747.6)
-135.0 (2.5)	-28.4 (10.5)	-10.9 (22.7)	7.7 (67.9)	268.9 (332.0)
error<50ms				69.56%
error<10ms				23.19%

(d) Non-Anchor Notes after Refinement

avg-error			15.4	(49.8)
std-dev			164.7	(157.8)
min-error			-517.1	(0.0)
max-error			1639.2	(1639.2)
-85.4 (1.0)	-9.6 (4.3)	-3.6 (8.6)	6.1 (22.7)	101.6 (240.7)
error<50ms				85.31%
error<10ms				57.01%

(e) Complete System

avg-error			14.4	(41.4)
std-dev			146.6	(141.4)
min-error			-539.0	(0.0)
max-error			1639.2	(1639.2)
-63.2 (0.7)	-7.3 (3.5)	-0.5 (7.6)	8.2 (17.6)	70.7 (196.7)
error<50ms				87.80%
error<10ms				60.97%

Table B.43.: Alignment Results k.533-3

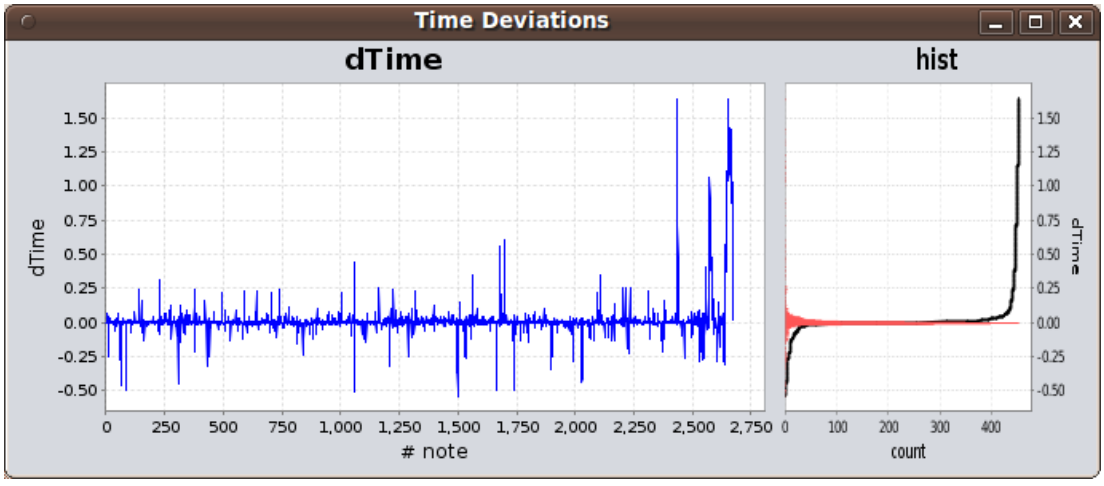


Figure B.39.: Time Deviations k.533-3

Bibliography

[Abdallah and Plumbley, 2004] Abdallah, S. A. and Plumbley, M. D. (2004). Polyphonic music transcription by non-negative sparse coding of power spectra. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, pages 318–325.

[Arifi et al., 2003] Arifi, V., Clausen, M., Kurth, F., and Müller, M. (2003). Automatic synchronization of music data in score-, midi- and pcm-format. In *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, Baltimore, MD, USA.

[Arzt and Widmer, 2010] Arzt, A. and Widmer, G. (2010). Towards effective "anytime" music tracking. In *Proceedings of the Starting AI Researchers' Symposium (STAIRS 2010)*.

[Arzt et al., 2008] Arzt, A., Widmer, G., and Dixon, S. (2008). Automatic page turning for musicians via real-time machine listening. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI 2008)*.

[Arzt et al., 2012] Arzt, A., Widmer, G., and Dixon, S. (2012). Adaptive distance normalization for real-time music tracking. In *Submitted to the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP2012)*.

[Atiquzzaman, 1994] Atiquzzaman, M. (1994). Complete line segment description using the hough transform. *Image and Vision Computing*, 12(5):267–273.

[Bartsch and Wakefield, 2001] Bartsch, M. A. and Wakefield, G. H. (2001). To catch a chorus: Using chroma-based representations for audio thumbnailing. In *Proceedings of the IEEE Workshop on Application of Signal Processing to Audio and Acoustics (WASPAA 2001)*, pages 15–18.

[Bayram and Selesnick, 2009] Bayram, I. and Selesnick, I. W. (2009). Frequency-domain design of overcomplete rational-dilation wavelet transforms. *IEEE Trans.*

Signal Processing (IEEE Transactions on Signal Processing), 57(8):2957–2972.

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

[Bertin et al., 2010] Bertin, N., Badeau, R., and Vincent, E. (2010). Enforcing harmonicity and smoothness in bayesian non-negative matrix factorization applied to polyphonic music transcription. *IEEE Transactions on Audio, Speech and Language Processing*, 18(3):538–549.

[Bertin-Mahieux et al., 2010] Bertin-Mahieux, T., Weiss, R. J., and Ellis, Daniel P. W. (2010). Clustering beat-chroma patterns in a large music database. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 111–116.

[Bloch and Dannenberg, 1985] Bloch, J. J. and Dannenberg, R. B. (1985). Real-time computer accompaniment of keyboard perfromances. In *Proceedings of the 11th International Computer Music Conference (ICMC 1985)*.

[Boashash, 1992] Boashash, B. (1992). Estimating and interpreting the instantaneous frequency of a signal: Part 1: Fundamentals. *Proceedings of the IEEE*, 80(4):520–538.

[Brossier, 2006] Brossier, P. M. (2006). *Automatic Annotation of Musical Audio for Interactive Applications*. PhD-Thesis, Queen Mary, University of London, UK.

[Brown and Puckette, 1992] Brown, J. C. and Puckette, M. S. (1992). An efficient algorithm for the calculation of a constant q transform. *Journal of the Acoustic Society of America*, 92(5).

[Böck and Widmer, 2012] Böck, S. and Widmer, G. (2012). Music transcription. In *Submitted to the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP2012)*.

[Cano et al., 1999] Cano, P., Loscos, A., and Bonada, J. (1999). Score-performance matching using hmms. In *Proceedings of the International Computer Music Conference (ICMC 1999)*.

[Cheveigné, 1993] Cheveigné, A. d. (1993). Separation of concurrent harmonic sounds: Fundamental frequency estimation and a time-domain cancellation model of auditory processing. *Journal of the Acoustic Society of America*, 93(6):3271–3290.

[Cont, 2006] Cont, A. (2006). Realtime audio to score alignment for polyphonic music instruments using sparse non-negative constraints and hierarchical hmms. In *Proceedings of the IEEE International Conference in Acoustics and Speech Signal Processing (ICASSP 2006)*.

[Cont, 2010] Cont, A. (2010). A coupled duration-focused architecture for real-time music-to-score alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):974—987.

[Cont et al., 2007] Cont, A., Schwarz, D., Schnell, N., and Raphael, C. (2007). Evaluation of real-time audio-to-score alignment. In *Proceedings of the 8^{th} International Conference on Music Information Retrieval (ISMIR 2007)*.

[Cooley and Tukey, 1965] Cooley, J. and Tukey, J. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301.

[Dannenberg, 1984] Dannenberg, R. B. (1984). An on-line algorithm for real-time accompaniment. In *Proceedings of the 10^{th} International Computer Music Conference (ICMC 1984)*.

[Dannenberg et al., 2007] Dannenberg, R. B., Birmingham, W., Pardo, B., Hu, N., Meek, C., and Tzanetakis, G. (2007). A comparative evaluation of search techniques for query-by-humming using the musart testbed. *Journal of the American Society for Information Science and Technology*, 58(5):687–701.

[Dannenberg and Hu, 2002] Dannenberg, R. B. and Hu, N. (2002). Discovering musical structure in audio recordings. In *Proceedings of the 2^{nd} International Conference on Music and Artificial Intelligence (ICMAI 2002)*.

[Dannenberg and Hu, 2003] Dannenberg, R. B. and Hu, N. (2003). Polyphonic audio mathcing for score following and intelligent audio editors. In *Proceedings of the International Computer Music Conference (ICMC 2003)*, pages 27–33.

[Dannenberg and Raphael, 2006] Dannenberg, R. B. and Raphael, C. (2006). Music score alignment and computer accompaniment. *Communications of the ACM*, 49(8):38.

[Daubechies, 1988] Daubechies, I. (1988). Orthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41(7):909—-996.

[Davies, 2007] Davies, Matthew E. P. (2007). *Towards Automatic Rhythmic Accompaniment*. PhD-Thesis, Queen Mary, University of London, UK.

[Devaney and Ellis, 2009] Devaney, J. and Ellis, Daniel P. W. (2009). Handling asynchrony in audio-score alignment. In *Proceedings of the International Computer Music Conference (ICMC 2009)*.

[Dixon, 2000] Dixon, S. (2000). On the computer recognition of solo piano musci. In *Proceedings of Australasian Computer Music Conference*, pages 31–37.

[Dixon, 2005a] Dixon, S. (2005a). Live tracking of musical performances using on-line time warping. In *Proceedings of the 8th International Conference on Digital Audio Effects (DAFx-05)*.

[Dixon, 2005b] Dixon, S. (2005b). An on-line time warping algorithm for tracking musical performances. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*.

[Dixon, 2006] Dixon, S. (2006). Onset detection revisited. In *Proceedings of the 9th International Conference on Digital Audio Effects 2006 (DAFx-06)*.

[Dixon et al., 2002] Dixon, S., Goebl, W., and Widmer, G. (2002). The performance worm: Real time visualisation based on langner's representation. In *Proceedings of the International Computer Music Conference (ICMC 2002)*, pages 361–364.

[Dixon and Widmer, 2005] Dixon, S. and Widmer, G. (2005). Match: A music alignment tool chest. In *Proceedings of the 6th International Conference on Music Infromation Retrieval (ISMIR 2005)*.

[Downie et al., 2010] Downie, J. S., Ehmann, A., Bay, M., and Jones, M. C. (2010). The music information retrieval evaluation exchange: Some observations and insights. In Raś, Z. W. and Wieczorkowska, A. A., editors, *Advances in Music Information Retrieval*, pages 93–115. Springer-Verlag Berlin Heidelberg, Berlin and Heidelberg, Germany.

[Dressein et al., 2010] Dressein, A., Cont, A., and Lemaitre, G. (2010). Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*.

[Dressler and Streich, 2007] Dressler, K. and Streich, S. (2007). Tuning frequency estimation using circular statistics. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*.

[Dörfler, 2002] Dörfler, M. (2002). *Gabor Analysis for a Class of Signals called Music*. PhD-Thesis, University of Vienna, Austria.

[Dörfler, 2004] Dörfler, M. (2004). What time-frequency analysis can do to music signals. In Emmer, M., editor, *Matematica e Cultura*. Springer, Italy.

[Ellis et al., 2008] Ellis, Daniel P. W., Cotton, C. V., and Mandel, M. I. (2008). Cross-correlation of beat-synchronous representations for music similarity. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2008)*, pages 57–60.

[Ewert and Müller, 2009] Ewert, S. and Müller, M. (2009). Refinement strategies for music synchronization. In *Lecture Notes on Computer Science*, volume 5493, pages 147–165. Springer, Berlin and Heidelber, Germany.

[Eyben et al., 2010] Eyben, F., Böck, S., Schuller, B., and Graves, A. (2010). Universal onset detection with bidirectional long short-term memory neural networks. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*.

[Flanagan and Golden, 1966] Flanagan, J. and Golden, R. (1966). Phase vocoder. *Bell Systems Technical Journal*, 45(9):1493–1509.

[Fletcher, 1964] Fletcher, H. (1964). Normal vibration frequencies of a stiff piano string. *The Journal of the Acoustical Society of America*, 36(6):203–209.

[Flossmann et al., 2010] Flossmann, S., Goebl, W., Grachten, M., Niedermayer, B., and Widmer, G. (2010). The Magaloff project: An interim report. *JNMR*, 39(4):363–377.

[Flossmann et al., 2011] Flossmann, S., Grachten, M., and Widmer, G. (2011). Expressive performance with bayesian networks and linear basis models. In *International Performance Rendering Contest RENCON 2011*.

[Flossmann and Widmer, 2011] Flossmann, S. and Widmer, G. (2011). Toward a multilevel model of expressive piano performance. In *International Symposium on Performance Science 2011*.

[Fremerey et al., 2010] Fremerey, C., Müller, M., and Clausen, M. (2010). Handling repeats and jumps in score-performance synchronization. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*.

[Friberg and Sundberg, 1992] Friberg, A. and Sundberg, J. (1992). Perception of just noticeable time displacement of a tone presented in a metrical sequence at different tempos. *STL-QPSR*, 33(4):97–108.

[Frigo and Johnson, 2005] Frigo, M. and Johnson, S. G. (2005). The design and implementation of fftw3. *Proceedings of the IEEE - Special Issue on Program Generation, Optimization, and Platform Adaption*, 93(2):216–231.

[Fujishima, 1999] Fujishima, T. (1999). Realtime chord recognition of musical sound: A system using common lisp music. In *Proceedings of the International Computer Music Conference (ICMC 1999)*, pages 464–467.

[Godsill and Davy, 2002] Godsill, S. and Davy, M. (2002). Bayesian harmonic models for musical signal analysis. In *7th Valencia International Meeting on Bayesian Statistics*.

[Goebl, 2001] Goebl, W. (2001). Melody lead in piano performance: Expressive device or artifact? *The Journal of the Acoustical Society of America*, 110(1):563–572.

[Goebl, 2003] Goebl, W. (2003). *The Role of Timing and Intensity in the Production and Perception of Melody in Expressive Piano Performance*. PhD-Thesis, Karl-Franzens University of Graz, Austria.

[Goebl and Bresin, 2003] Goebl, W. and Bresin, R. (2003). Measurement and reproduction accuracy of computer-controlled grand pianos. In *Proceedings of the Stockholm Music Acoustics Conference (SMAC 2003)*.

[Goto, 2005] Goto, M. (2005). Prefest: A predominant-f0 estimation method for polyphonic musical audio signals. In *Proceedings of the 2nd Music Information Retrieval Evaluation eXchange (MIREX 2005)*.

[Goto, 2006] Goto, M. (2006). A chorus section detection method for musical audio signals and its application to a music listening station. *IEEE Transactions on Audio, Speech and Language Processing*, 14(5):1783–1794.

[Goto and Hayamizu, 1999] Goto, M. and Hayamizu, S. (1999). A real-time music scene description system: Detecting melody and bass lines in audio signals. In *Proceedings of the IJCAI-99 Workshop on Computational Auditory Scene Analysis*, pages 31–40.

[Grachten and Widmer, 2011] Grachten, M. and Widmer, G. (2011). Explaining musical expression as a mixture of basis functions. In *Proceedings of the 8^{th} Sound and Music Computing Conference (SMC 2011).*

[Gómez, 2006] Gómez, E. (2006). *Tonal Description of Music Audio Signals.* PhD-Thesis, Universitat Pompeu Fabra, Barcelona, Spain.

[Hainsworth, 2001] Hainsworth, S. W. (2001). *Analysis of Musical Audio for Polyphonic Transcription.* PhD thesis – 1^{st} year reoport, University of Cambridge, Cambridge, UK.

[Hainsworth, 2003] Hainsworth, S. W. (2003). *Techniques for Audio Transcription.* PhD-Thesis, University of Cambridge, Cambridge, UK.

[Han and Raphael, 2010] Han, Y. and Raphael, C. (2010). Informed source separation of orchestra and soloist. In *Proceedings of the 11^{th} International Society for Music Information Retrieval Conference (ISMIR 2010).*

[Heideman et al., 1984] Heideman, M. T., Johnson, D. H., and Burrus, C. S. (1984). Gauss and the history of the fast fourier transform. *IEEE ASSP Magazin*, 1(4):14–21.

[Hu and Dannenberg, 2005] Hu, N. and Dannenberg, R. B. (2005). A bootstrap method for training an accurate audio segmenter. In *Proceedings of the 6^{th} International Conference on Music Infromation Retrieval (ISMIR 2005).*

[Hu and Dannenberg, 2006] Hu, N. and Dannenberg, R. B. (2006). Bootstrap learning for accurate onset detection. *Machine Learning*, 65(2-3):457–471.

[Hu et al., 2003] Hu, N., Dannenberg, R. B., and Tzanetakis, G. (2003). Polyphonic audio matching and alignment for music retrieval. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2003)*, pages 185–188.

[Jiang et al., 2011] Jiang, N., Grosche, P., Konz, V., and Müller, M. (2011). Analyzing chroma feature types for automated chord recognition. In *Proceedings of the 42^{nd} AES Conference.*

[Joder et al., 2010a] Joder, C., Essid, S., and Richard, G. (2010a). A comparative study of tonal acoustic features for a symbolic level music-to-score alignment. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2010).*

[Joder et al., 2010b] Joder, C., Essid, S., and Richard, G. (2010b). An improved hierarchical approach for music-to-symbolic score alignment. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010).*

[Kapur et al., 2004] Kapur, A., Benning, M., and Tzanetakis, G. (2004). Query-by-beat-boxing: Music retrieval for the dj. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004).*

[Klapuri, 1999] Klapuri, A. (1999). Pitch estimation using multiple independent time-frequency windows. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 1999).*

[Klapuri, 2003] Klapuri, A. (2003). Multiple fundamental frequency estimation based on harmonicity and spectral smoothness. *IEEE Transactions on Speech and Audio Processing*, 11(6):804–816.

[Langner and Goebl, 2003] Langner, J. and Goebl, W. (2003). Visualizing expressive performance in tempo-loudness space. *Computer Music Journal*, 79(4):69–83.

[Langner et al., 2000] Langner, J., Kopiez, R., and Stoffel, C. (2000). Realtime analysis of dynamic shaping. In *Proceedings of the 6th International Conference on Music Perception and Cognition.*

[Lawson and Hanson, 1974] Lawson, C. and Hanson, R. (1974). *Solving Least Squares Problems.* Prentice Hall, Lebanon, IN, USA.

[Lee and Seung, 1999] Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

[Lee and Seung, 2001] Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In *13th Conference on Advances in Neural Information Processing Systems*, page 556–562. MIT Press, Denver, CO, USA.

[Lerch, 2006] Lerch, A. (2006). On the requirement of automatic tunig frequency estimation. In *Proceedings of the 7th International Conference on Music Information*

Retrieval (ISMIR 2006).

[Liu et al., 2010] Liu, Y., Dannenberg, R. B., and Cai, L. (2010). The intelligent music editor: Towards an automated platform for music analysis and editing. In Huang, D.-S., Zhang, X., García, C. A., and Zhang, L., editors, *Advanced Intelligent Computing Theories and Applications*, volume 6216 of *Lecture Notes in Computer Science*, pages 123–131. Springer, Berlin and Heidelberg, Germany.

[Lu et al., 2004] Lu, L., Wang, M., and Zhang, H. (2004). Repeating pattern discovery and structure analysis from acoustic music data. In *Proceedings of the 6th ACM SIGMM International Workshop on Multimedia Information Retrieval.*

[Macrae and Dixon, 2010] Macrae, R. and Dixon, S. (2010). Accurate real-time windowed time warping. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 423–428.

[Mallat, 1989] Mallat, S. G. (1989). A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7).

[Martin, 1996] Martin, K. D. (1996). *A Blackboard System for Automatic Transcription of Simple Polyphonic Music.* PhD thesis, MIT, MA, USA.

[Meron and Hirose, 2001] Meron, Y. and Hirose, K. (2001). Automatic alignment of a musical score to performed music. *Acoustical Science & Technology*, 22(3):189–198.

[Moog and Rhea, 1990] Moog, R. A. and Rhea, T. L. (1990). Evolution of the keyboard interface: The bösendorfer 290 se recording piano and themoog multiply-touch-sensitive keyboards. *Computer Music Journal*, 14(2):52–60.

[Müller, 2011] Müller, M. (2011). Evaluation issues in the field of audio-to-score alignment: Personal communication.

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

[Müller et al., 2005] Müller, M., Kurth, F., and Clausen, M. (2005). Audio matching via chroma-based statistical features. In *Proceedings of the 6th International Conference on Music Infromation Retrieval (ISMIR 2005).*

[Müller et al., 2004] Müller, M., Kurth, F., and Röder, T. (2004). Towards an efficient algorithm for automatic score-to-audio synchronization. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, pages 365–372.

[Müller et al., 2006] Müller, M., Mattes, H., and Kurth, F. (2006). An efficient multiscale approach to audio synchronization. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, pages 192–197.

[Niedermayer, 2008] Niedermayer, B. (2008). Non-negative matrix division for the automatic transcription of polyphonic music. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*.

[Niedermayer, 2009a] Niedermayer, B. (2009a). Improving accuracy of polyphonic music-to-score alignment. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*.

[Niedermayer, 2009b] Niedermayer, B. (2009b). Towards audio to score alignment in the symbolic domain. In *Proceedings of the 6th Sound and Music Computing Conference (SMC 2009)*.

[Niedermayer et al., 2011a] Niedermayer, B., Böck, S., and Widmer, G. (2011a). On the importance of "real" audio data for MIR algorithm evaluation at the note-level – a comparative study. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*.

[Niedermayer and Widmer, 2010a] Niedermayer, B. and Widmer, G. (2010a). A multi-pass algorithm for accurate audio-to-score alignment. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*.

[Niedermayer and Widmer, 2010b] Niedermayer, B. and Widmer, G. (2010b). Strategies towards the automatic annotation of classical piano music. In *Proceedings of the 7th Sound and Music Computing Conference (SMC 2010)*.

[Niedermayer et al., 2011b] Niedermayer, B., Widmer, G., and Reuter, C. (2011b). Version detection for historical musical automata. In *Proceedings of the 8th Sound and Music Computing Conference (SMC 2011)*.

[Ong et al., 2006] Ong, B. S., Gómez, E., and Streich, S. (2006). Automatic extraction of musical structure using pitch class distribution features. In *Workshop on Learning*

the Semantics of Audio Signals (LSAS 2006), pages 53–65.

[Oppenheim and Schafer, 2007] Oppenheim, A. V. and Schafer, R. W. (2007). *Discrete-Time Signal Processing*. Prentice Hall, Harlow, UK, 3^{rd} international ed. edition.

[Orio and Déchelle, 2001] Orio, N. and Déchelle, F. (2001). Score following using spectral analysis and hidden markov models. In *Proceedings of the International Computer Music Conference (ICMC 2001)*, pages 151–154.

[Painter and Spanias, 2000] Painter, T. and Spanias, A. (2000). Perceptual coding of digital audio. *Proceedings of the IEEE*, 88(4):451–515.

[Pardo and Sanghi, 2005] Pardo, B. and Sanghi, M. (2005). Polyphonic musical sequence alignment for database search. In *Proceedings of the 6^{th} International Conference on Music Infromation Retrieval (ISMIR 2005)*, pages 215–222.

[Patterson et al., 1992] Patterson, R. D., Robinson, K., Holdsworth, J., McKeown, D., Zhang, C., and Allerhand M. (1992). Complex sounds and auditory images. In *Auditory physiology and perception, Proceedings of the 9th International Symposium on Hearing*, pages 429–446. Pergamon, Oxford, UK.

[Peeters et al., 2002] Peeters, G., La Burthe, A., and Rodet, X. (2002). Roward automatic music audio summary generation from signal analysis. In *Proceedings of the 3^{rd} International Symposium on Music Information Retrieval (ISMIR 2002)*.

[Plumbley et al., 2006] Plumbley, M. D., Abdallah, S. A., Blumensath, T., and Davies, M. E. (2006). Sparse representations of polyphonic music. *Signal Processing*, 86(3):417–431.

[Poliner and Ellis, 2007] Poliner, G. E. and Ellis, Daniel P. W. (2007). A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, 2007(1):1–10.

[Princen et al., 1992] Princen, J., Illingworth, J., and Kittler, J. (1992). A formal definition of the hough transform - properties and relationships. *Journal of Mathematical Imaging and Vision*, 1(2):153–168.

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

[Rabiner and Juang, 1993] Rabiner, L. R. and Juang, B.-H. (1993). *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, NJ, USA.

[Raphael, 1999] Raphael, C. (1999). Automatic segmentation of acoustic musical signals using hidden markov models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):360–380.

[Raphael, 2001] Raphael, C. (2001). Coarse-to-fine dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1379–1390.

[Raphael, 2006] Raphael, C. (2006). Aligning music audio with symbolic scores using a hybrid graphical model. *Machine Learning Journal*, 65(2):389–409.

[Raphael, 2009] Raphael, C. (2009). Current directions with musical plus one. In *Proceedings of the 6th Sound and Music Computing Conference (SMC 2009)*.

[Ravuri and Ellis, 2010] Ravuri, S. and Ellis, Daniel P. W. (2010). Cover song detection: From high scores to general classification. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2010)*, pages 65–68.

[Ryynänen and Klapuri, 2005] Ryynänen, M. and Klapuri, A. (2005). Polyphonic music transcription using note event modeling. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2005)*.

[Sakoe and Chiba, 1978] Sakoe, H. and Chiba, S. (1978). Dynamic programming optimization for spoken word recognition. *IEEE Trans. Acoustics, Speech, Sig. Processing (IEEE Transactions on Acoustics, Speech, and Signal Processing)*, 26(1):43–49.

[Salvador and Chan, 2004] Salvador, S. and Chan, P. (2004). Fastdtw: Toward accuarte dynamic time warping in linear time and space. In *KDD Workshop on Mining Temporal and Sequential Data*, pages 70–80.

[Salvador and Chan, 2007] Salvador, S. and Chan, P. (2007). Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580.

[Scheirer, 1997] Scheirer, E. D. (1997). Using musical knowledge to extract expressive performance information from audio recordings. In Okuno, H. G. and Rosenthal, D., editors, *Readings in Computational Auditory Scene Analysis*, pages 361–380. Lawrence Erlbaum Publication, Mahew, NJ, USA.

[Schuck and Young, 1943] Schuck, O. and Young, R. (1943). Observations on the vibrations of piano strings. *The Journal of the Acoustical Society of America*, 15(1):1–11.

[Schwarz et al., 2004] Schwarz, D., Orio, N., and Schnell, N. (2004). Robust polyphonic midi score following with hidden markov models. In *Proceedings of the International Computer Music Conference (ICMC 2004)*, pages 3–6.

[Schörkhuber and Klapuri, 2010] Schörkhuber, C. and Klapuri, A. (2010). Constant-q transform toolbox for music processing. In *Proceedings of the 7^{th} Sound and Music Computing Conference (SMC 2010)*.

[Serrà et al., 2010] Serrà, J., Gómez, E., and Herrera, P. (2010). Audio cover song identification and similarity: Background, approaches, evaluation, and beyond. In *Advances in Music Information Retrieval*, volume 247 of *Studies in Computational Intelligence*, pages 307–332. Springer, Berlin, Germany.

[Serrà et al., 2008] Serrà, J., Gómez, E., Herrera, P., and Serra, X. (2008). Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(6):1138–1151.

[Serrà et al., 2009] Serrà, J., Serra, X., and Andrzejak, R. G. (2009). Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11(9):093017.

[Shalev-Schwartz et al., 2004] Shalev-Schwartz, S., Keshet, J., and Singer, Y. (2004). Learning to align polyphonic music. In *Proceedings of the 5^{th} International Conference on Music Information Retrieval (ISMIR 2004)*.

[Sheh and Ellis, 2003] Sheh, A. and Ellis, Daniel P. W. (2003). Chord segmentation and recognition using em-trained hidden markov models. In *Proceedings of the 4^{th} International Conference on Music Information Retrieval (ISMIR 2003)*, pages 185–191, Baltimore, MD, USA.

[Shiu et al., 2006] Shiu, Y., Jeong, H., and Jay Kuo, C.-C. (2006). Similarity matrix processing for music structure analysis. In *Proceedings of the Audio and Music Computing for Multimedia Workshop (AMCMM 2006)*.

[Smaragdis and Brown, 2003] Smaragdis, P. and Brown, J. C. (2003). Non-negative matrix factorization for polyphonic music transcription. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2003)*.

[Song et al., 2002] Song, J., Bae, S.-Y., and Yoon, K. (2002). Query by humming: Matching humming query to polyphonic audio. In *Proceedings of the 2002 IEEE International Conference on Multimedia and Expo*, pages 329–332.

[Soulez et al., 2003] Soulez, F., Rodet, X., and Schwarz, D. (2003). Improving polyphonic and poly-instrumental music to score alignment. In *Proceedings of the 4^{th} International Conference on Music Information Retrieval (ISMIR 2003)*, Baltimore, MD, USA.

[Turetsky and Ellis, 2003] Turetsky, R. J. and Ellis, Daniel P. W. (2003). Ground-truth transcriptions of real music from force-aligned midi syntheses. In *Proceedings of the 4^{th} International Conference on Music Information Retrieval (ISMIR 2003)*, Baltimore, MD, USA.

[Tzanetakis et al., 2003] Tzanetakis, G., Emolinskyi, A., and Cook, P. (2003). Pitch histograms in audio and symbolic music information retrieval. *JNMR*, 32(2):143–152.

[Tzanetakis et al., 2001] Tzanetakis, G., Essl, G., and Cook, P. (2001). Audio analysis using the discrete wavelet transform. In *Proceedings of the WSES International Conference on Acoustics and Music: Theory and Applications (AMTA 2001)*.

[Urbano, 2011] Urbano, J. (2011). Information retrieval meta-evaluation: Challenges and opportunities in the music domain. In *Proceedings of the 12^{th} International Society for Music Information Retrieval Conference (ISMIR 2011)*.

[Velasco et al., 2011] Velasco, G. A., Holighaus, N., Dörfler, M., and Grill, T. (2011). Constructing an invertible constant-q transform with nonstationary gabor frames. In *Proceedings of the 14^{th} International Conference on Digital Audio Effects (DAFx-11)*, pages 93–99.

[Vincent et al., 2007] Vincent, E., Bertin, N., and Badeau, R. (2007). Two nonnegative matrix factorization methods for polyphonic pitch transcription. In *MIREX 2007*.

[Vincent et al., 2008] Vincent, E., Bertin, N., and Badeau, R. (2008). Harmonic and inharmonic nonnegative matrix factorization for polyphonic pitch transcription. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2008)*.

[Virtanen, 2007] Virtanen, T. (2007). Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE Transactions on Audio, Speech and Language Processing*, 15(3):1066–1074.

[Virtanen et al., 2008] Virtanen, T., Cemgil, A. T., and Godsill, S. (2008). Bayesian extensions to non-negative matrix factorisation for audio signal modelling. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2008)*.

[Weiss and Bello Correa, 2010] Weiss, R. J. and Bello Correa, J. P. (2010). Identifying repeated patterns in music using sparse convolutive non-negative matrix factorization. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 123–128.

[Widmer et al., 2009] Widmer, G., Flossmann, S., and Grachten, M. (2009). Yqx plays chopin. *AI Magazine*, 30(3):35–48.

curriculum vitæ

Dipl.-Ing. Mag.
Bernhard Niedermayer

b.niedermayer@inode.at

Date of Birth	July 28st 1983
Nationality	Austrian
Address	Colerusstr. 12/9
	A-4040 Linz

Education and Studies

1992–2001	Grammar School
2001–2005	Bachelor's programme Computer Science at the Johannes Kepler University (JKU) of Linz (Austria)
2005	Exchange programme at the Oxford Brookes University (UK)
2005–2009	Master's programme Business Information Systems at the Johannes Kepler University of Linz
2006–2007	Master's programme Computer Science at the Johannes Kepler University of Linz
2007–	PhD programme Computer Science at the Johannes Kepler University of Linz
2009–	Master programme Legal and Business Aspects in Technics at the Johannes Kepler University of Linz

Practial experience

2004–2005	Developer of eLearning material at the Institute for Information Processing and Microprocessor Technology (JKU)
2004–2006	Tutor at the Institute for Applied Knowledge Processing (JKU)
2006	Developer of eLearning material at the Oxford Brookes University
2006–2007	IT-Trainer at the company Integanet (Software Development)
2008	IT-Trainer at the company Rossi Roth KG (Software Development)
2007–2008	IT Administrator and Advisor at the Ziviltechnikerbuero Steinbichl
2007–	Research Assistant at the Department for Computational Perception (JKU)

Publications

2007	Niedermayer, B. *Automatic Detection of Cover-Versions and Plagiarism in Modern Pop- und Rockmusic.* Master-Thesis (in German), JKU Linz.
2008	Niedermayer, B. *Non-Negative Matrix Division for the Automatic Transcription of Polyphonic Music.* In Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008), Philadelphia, PA, USA.
2009	Niedermayer, B. *Towards Audio to Score Alignment in the Symbolic Domain.* In Proceedings of the Sound and Music Computing Conference (SMC 2009), Porto, Portugal.
2009	Niedermayer, B. *Improving Accuracy of Polyphonic Music-to-Score Alignment.* In Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR 2009), Kobe, Japan.
2010	Niedermayer, B.; Widmer, G. *A Multi-Pass Algorithm for Accurate Audio-to-Score Alignment.* In Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010), Utrecht, The Netherlands.
2010	Niedermayer, B.; Widmer, G. *Strategies towards the Automatic Annotation of Classical Piano Music.* In Proceedings of the 7th Sound and Music Computing Conference (SMC 2010), Barcelona, Spain.
2010	Flossmann, S.; Goebl, W.; Grachten, M.; Niedermayer, B.; Widmer, G. *The Magaloff Project: An Interim Report.* Journal of New Music Research, 39 (4), 363-377 .
2011	Niedermayer, B.; Widmer, G.; C. Reuter *Version Detection for Historical Musical Automata.* In Proceedings of the 8th Sound and Music Computing Conference (SMC 2011), Padova, Italy.
2011	Niedermayer, B.; Böck, S; Widmer, G. *On the Importance of "Real" Audio Data for MIR Algorithm Evaluation at the Note-Level - A comparative Study.* In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), Miami, Florida, USA.

Linz, February 28, 2012

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Linz, am _____

Bernhard Niedermayer