Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science

Master's Thesis

A Cross-Version Approach for Novelty Detection in Music Recordings

submitted by
Thomas Prätzlich

submitted
December 22, 2011

Supervisor / Advisor
Priv.-Doz. Dr. Meinard Müller

Reviewers
Priv.-Doz. Dr. Meinard Müller
Prof. Dr. Michael Clausen

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement under Oath

I confirm under oath that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, _____ _____

(Datum / Date) (Unterschrift / Signature)

Acknowledgements

First, I want to thank my supervisor Meinard Müller for supervising my thesis, and for always directing me back into the right direction.

Special thanks go to Jonathan Driedger, who was always available for inspiring discussions, and who spent quite a significant amount of time with proof reading my thesis. I would also like to thank Peter Ebert, Sebastian Finkel, and Nanzhu Jiang for proof reading as well. To Nanzhu, my thanks also go for annotating most of the pieces in the dataset, as well as to Verena Konz, who also gave me inspiring feedback and made significant contributions on MetaMIDI. I would like to thank Peter Grosche for his feedback which was always much appreciated.

I also want to thank the people that I met at USMIR2010 (the summer school of ISMIR 2010) for the inspiring discussions and the good times we had: Hendrik Schreiber, Tom Collins, Anastassia Semjonova.

I also want to express my deep gratitude to Geraint Wiggins, who let me do research in his group during my ERASMUS year. Also, I want to thank all the people that I met in his group and the people that I met at Goldsmiths College. I want to thank Annika Gräwe who always gave me a place of calm and useful advice on the use of the English language. Last but not least, I want to thank my family who has always supported me.

Abstract

Music is highly structured data. Structure in music arises from repetitions, contrasts and homogeneity in musical aspects such as melody, dynamics, harmony, timbre or tempo. The extraction of the musical structure from audio recordings is an important task in the field of music information retrieval. It consists of a segmentation problem, where the goal is to find the boundaries which mark the transitions between two structural parts, and a musically meaningful labeling of the segments, e.g. chorus or verse. Typically, segment boundaries are accompanied by a change in one or more musical aspects. The task of novelty detection is to find those points of change.

In this thesis, we deal with the task of novelty detection in music recordings. Here, we first convert the audio signal into a suitable feature sequence to compute a self distance matrix on it from which a novelty curve is derived. To model the progression of the musical aspects harmony, timbre and tempo in a recording, we use chroma, MFCC and cyclic tempogram features. Particularly in classical music, there often exist many recordings which are performed according to a given musical score. Although they share the same underlying structure provided by the score, two different recordings usually vary in performance aspects such as tempo, dynamics or timbre. This is the reason why the result of a structure analysis often varies between different performances. The main contribution of this thesis is to apply the novelty detection simultaneously to a set of different performances of a given piece. To this end, we apply dynamic time warping with a MIDI reference version to transform the novelty curve onto a common musical time axis. Then we combine the individual curves into a fusion curve. Our hypothesis is, that musically relevant points of novelty are consistent across different performances and therefore we expect the fusion curve to be more robust and musically meaningful than the individual curves. Our experiments show that on average, the fusion curve yields better results than the novelty curves of the single recordings. This effect becomes more prominent, when there is a high variance in the set of individual novelty curves.

Contents

"Without music life would be a mistake."

Friedrich Wilhelm Nietzsche

1 Introduction

One of the attributes distinguishing music from random sound sources is the structure in which music is organized in a hierarchical way. On a higher level, *musical form* provides an abstract description of a piece of music. Musical form is build from smaller structures such as sections, phrases and motifs. An example of musical form is the division into parts like verse or chorus in popular music. There are two basic principles of structure in music, namely the *temporal ordering* of musical elements and the organization of *simultaneously* occurring musical elements. For example, simultaneously sounding notes create harmonies, whereas the temporal ordering of harmonies creates the harmonic progression of the music. Based on the temporal ordering, one can identify additional principles like *repetition*, *contrast*, *homogeneity* or *variation* which induce structure in music [24]. Repetition is a central principle in music. Huron and Ollen studied a cross-cultural sample of fifty musical works and found that *"94 percent of all musical passages longer than a few seconds in duration are repeated at some point in the work"* [14]. In sonatas for example, the theme is repeated in the recapitulation part. In Beethoven's Fifth, the "fate motif" is repeated not only melodically, but also rhythmically. Repetitions evoke familiarity and understanding of the underlying musical material in the listener [24]. The principle of *contrast* applies to musical parts which differ in a certain musical aspect. For example, two parts of a musical work which are in different key induce a contrast. *Variations* are a special case of repetitions, where a part is not only repeated but also transformed. As already stated implicitly by the contrast principle, there also are sections which are *homogeneous* with regard to a certain musical aspect.

The extraction of musical structure is a central task in the field of music information retrieval (MIR). Ordered from lower hierarchical levels to high-level structures in music, pitch tracking, chord recognition and finally music structure analysis are examples of different tasks in MIR which all aim at extracting some kind of structure. Pitch tracking aims at dividing an audio stream into individual pitches, chord recognition assigns chord labels to regions which are consistent in harmony, and music structure analysis aims at

revealing the musical form. Technically, structure extraction consists of two subproblems: segmentation and labeling/classification. The goal of the segmentation task is to detect a set of boundaries which mark the transitions between consecutive segments, and the labeling aims at assigning a musically meaningful tag to each of the segments. There are many different methods for structure extraction, which can be divided into *repetition-based*, *homogeneity-based* and *novelty-based* approaches [24]. *Repetition-based* methods aim at finding recurring patterns in one or more musical aspects. The goal of *homogeneity-based* approaches is finding regions in a music recording which are consistent with respect to one or more musical aspects. Segment boundaries usually go along with a change in at least one musical aspect. The objective of *novelty-based* methods is to detect those points of change between two contrasting contiguous segments.

In this thesis, we focus on novelty detection in music recordings. To this end, we first need to transform the audio signal into suitable feature sequences which correlate with certain musical aspects. To account for the progressions in the musical aspects of harmony, timbre, and tempo, we use chroma, MFCC and tempogram features, respectively. In a further step, we apply an algorithm which is based on the analysis of a self-similarity matrix of a sequence to compute a novelty curve [7]. A novelty curve indicates points of change in a music recording. In the case of structural segmentation, the novelty curve serves as an indicator of segment boundaries. The curve exhibits high values at points of change and low values within homogeneous regions in a music recording. An example of a novelty curve resulting from the analysis of coarse harmonic changes in a recording of Chopin's Mazurka Op. 4 No. 4 is shown in Figure 1.1. The figure also shows the chroma feature sequence and the waveform which were used in the novelty computation. A structural segmentation is indicated on the physical time axis in seconds as well as in the musical score. In the feature representation, we can already see a lot of structure of the recording, whereas the waveform on its own does not yield much structural insights. Note that in the chroma feature sequence there are actually more structural patterns visible than peaks in the novelty curve. This is because the novelty curve computation was parametrized to capture rather coarse harmonic changes. One can see, that the novelty curve closely correlates with the annotated segment boundaries.

Since the early beginnings of recording technology for music, a vast amount of music has been recorded. In the recent history, the availability of music recordings in digital form has increased drastically. Especially for classical music, there often exist many different recordings for the same piece of music. These recordings share a common structure determined by the score. Nevertheless, the structure analysis of a music recording does not only depend on the abstract structure of the underlying piece but also on performance-specific characteristics. For example, one musician may emphasize a certain transition by playing a ritardando while another musician may change the dynamics. The main idea of this thesis is to apply the novelty detection algorithm simultaneously to a set of recordings of the same piece of music. To be able to compare the novelty curves of different recordings, we transform the curves onto a common musical time axis by applying *dynamic time warping* with a MIDI reference version. Our main hypothesis is that musically relevant points of novelty are consistent across different recordings, although the individual recordings vary in performance aspects. We analyze whether the results of the novelty detection can be improved by combining the individual novelty curves, creating thus a fusion curve.

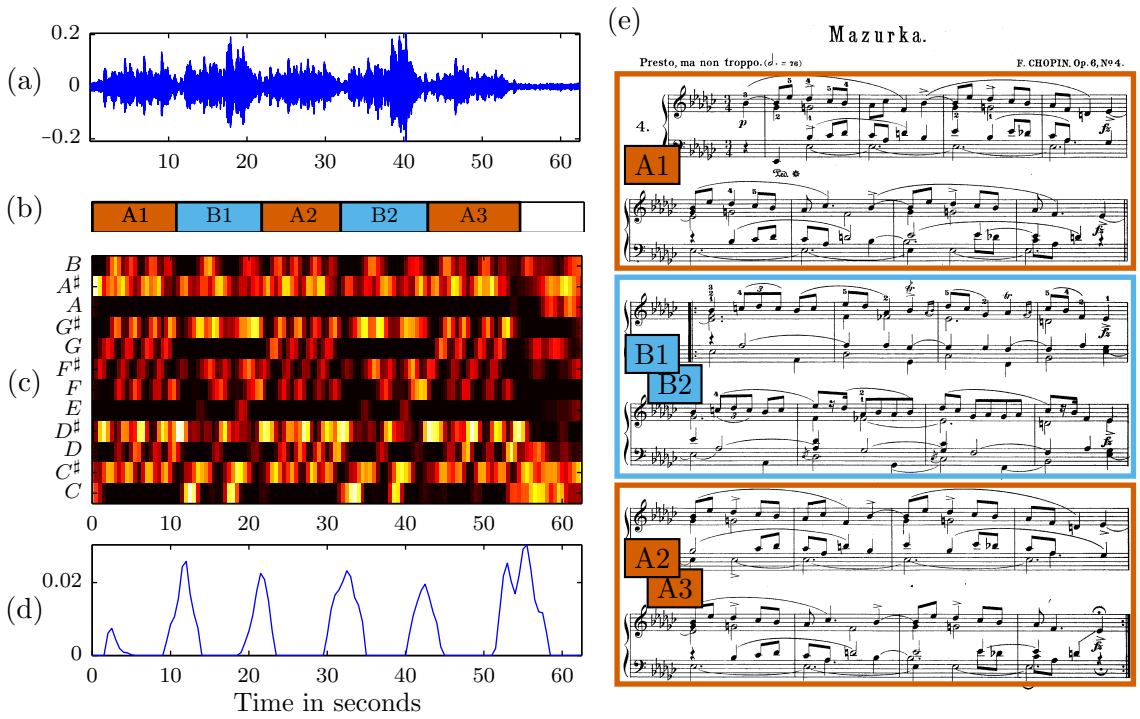


Figure 1.1. Illustration of novelty detection on a music recording. **(a)**: Waveform. **(b)**: Annotation on physical time axis in seconds. **(c)**: Chroma feature sequence. **(d)**: Novelty curve. **(e)**: Musical score with annotations corresponding to (b). Note that the parts B2 and A3 are repetitions of B1 and A2.

The basic pipeline of the cross-version approach for novelty detection is sketched in Figure 1.2. It shows a set of waveforms on which the novelty detection algorithm is applied. The novelty curves all have a different length, as the total duration of each recording is different. Then, the novelty curves are transformed onto the musical time axis using dynamic time warping with a MIDI version. After this transformation, one can see that most of the curves have a similar shape. In the last step, a fusion curve which combines the information of all curves is computed.

1.1 Related Work and Applications

Work which takes a similar direction as the cross-version approach introduced in the last section has been conducted on the analysis of chord recognition and beat tracking algorithms. Konz, Müller and Ewert developed a multi-perspective evaluation framework for chord recognition which uses music synchronization techniques to warp chord recognition results of different performances of the same piece onto a common musical time axis [16]. The results can then be compared to a ground truth annotation which was created by a musician on the basis of the underlying score of the piece. This approach has two advantages: First, it is more intuitive for a musician to produce annotations on the basis of a musical score than on an audio recording, and second, one needs only one annotation for

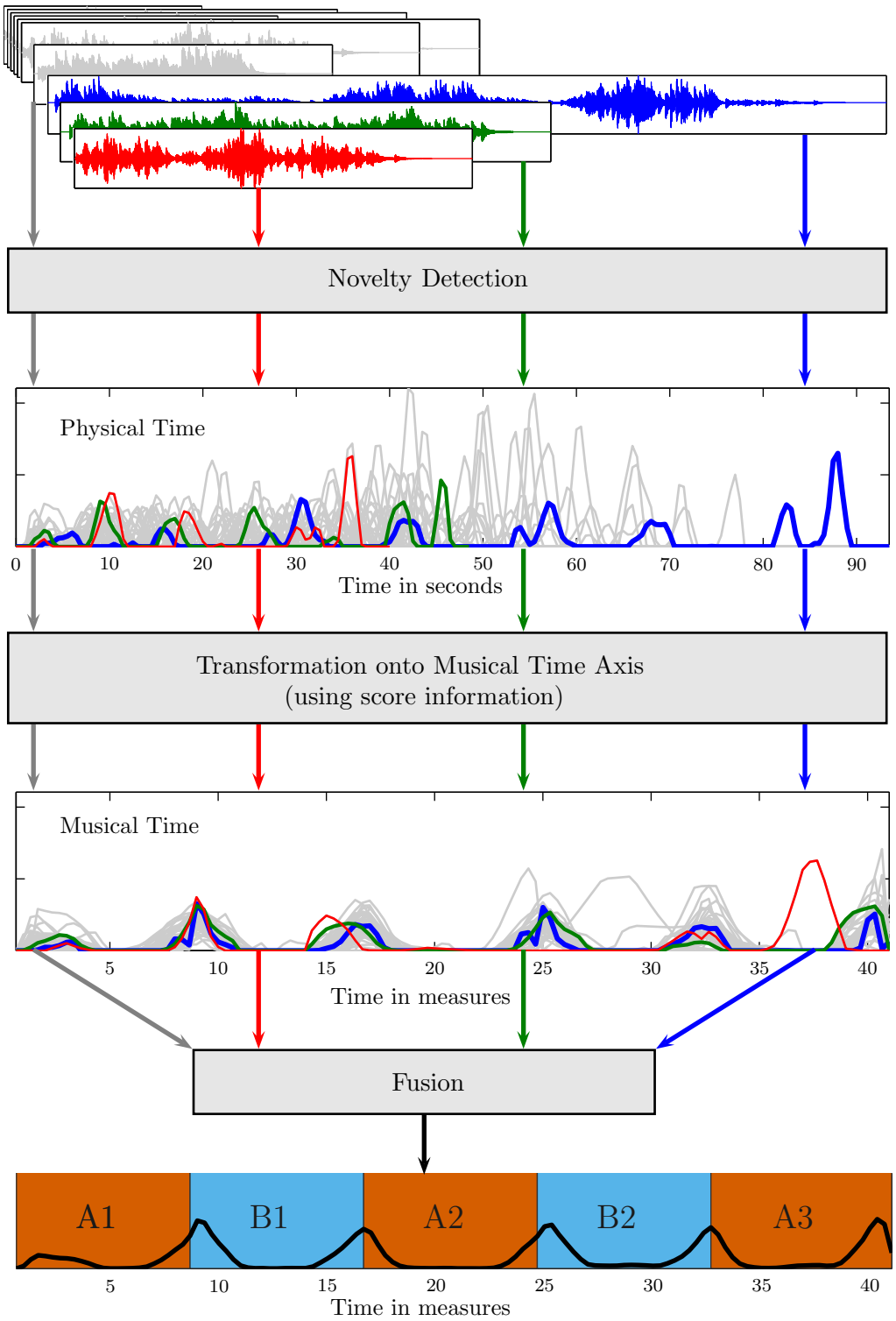


Figure 1.2. Schematic overview of the cross-version pipeline on Chopin's Mazurka Op. 4 No. 4. The corresponding musical score is shown in Figure 1.1e. Note that the background of the fusion curve (bottom of this figure) is a structural reference segmentation of this piece of music.

a set of recordings as it is not necessary to annotate every single recording on the physical time axis. A further contribution of their work is a visualization which can be used to find inconsistencies in the chord labeling procedure between different versions of a piece. One can then distinguish between errors which are inherent to ambiguities in the chord labeling or caused by a general weakness of the chord labeling algorithm. Their framework makes it easier to find outliers and to identify either musical reasons or general problems in a chord recognition algorithm.

Grosche, Müller and Sapp conducted a study on the difficulties in beat tracking, where they simultaneously determined consistencies of beat tracking errors over many performances [12]. The underlying assumption of their work was "that tracking errors consistently occurring in many performances of a piece are likely caused by musical properties of the piece rather than the physical properties of a specific performance". This is very similar to our hypothesis, but from an opposite point of view. Recall that we assumed that musically relevant points of novelty are consistent across different recordings and therefore inherent to the piece of music whereas Grosche et al. are looking at consistent errors which are inherent to musical properties in the piece. In general, one can look at consistencies to identify errors or to check whether an algorithm is operating correctly. An implementation of these ideas has been realized in an interface referred to as the "Interpretation Switcher" [21]. The Interpretation Switcher is an audio player which allows a user to select a set of recordings of the same piece of music for which an offline synchronization has been computed. The synchronization information is used to provide several functionalities. First, the user can switch seamlessly between the different recordings during the playback. Additionally, the application provides the possibility to display different annotations for each recording and for transforming them onto a common time axis for the purpose of comparison. Furthermore, the Interpretation Switcher makes it possible to look at consistencies and inconsistencies in the results of an algorithm applied to many different performances.

1.2 Thesis Organization

In the following, we give an outline of this thesis and briefly describe the content of each chapter.

Chapter 2 introduces basic concepts from signal processing. Furthermore, it provides a description of different audio features which correlate with the musical aspects of harmony, timbre, and tempo.

Chapter 3 describes the music synchronization pipeline based on the dynamic time warping.

Chapter 4 presents an algorithm to compute novelty curves from music recordings which is based on a self-distance matrix of a music recording.

Chapter 5 introduces evaluation metrics and our experimental setup. It furthermore explains how results of MIR tasks can be evaluated on a common musical time axis.

Chapter 6 presents the cross-version approach which builds upon the techniques introduced in the previous chapters. It describes how a set of novelty curves is transformed

onto a common musical time axis and introduces a visualization for the simultaneous analysis of different novelty curves. Furthermore, the computation of a fusion curve from a set of curves is explained and possible error sources in the approach are discussed.

Chapter 7 describes experiments on different levels of the cross-version pipeline.

Chapter 8 draws conclusions and points out directions of possible future research.

"You must learn to cherish what's implicit in things."

Heinz von Foerster

2 Feature Extraction

Music consists of several aspects such as pitch, harmony, rhythm, loudness or timbre. In music recordings, those aspects are implicitly encoded in the waveform. The transformation of the waveform into a musically more relevant representation is called *feature extraction*. The choice of features depends on the task one wants to solve. The audio feature should carry as much information as possible relevant to a certain task while being invariant to irrelevant aspects. For example, an audio feature for modeling the harmonic content should be invariant to timbral aspects. Features can be divided in *low-level*, *mid-level* and *high-level* features depending on their degree of abstraction. An example for a low-level feature is the frequency spectrum, which represents the frequency content of a signal. Mid-level features usually correlate with musical aspects. An example for this are chroma feature sequences (introduced in Section 2.2) which correlate with the harmonic progression in a piece of music. High-level features describe more abstract musical concepts like the musical key or instrumentation. It is important to differentiate clearly between musical aspects and audio features. Audio features are used to model a certain musical aspect within a specific task. For example, chroma features can be used for the detection of segment boundaries between sections which differ in the musical aspect harmony. However, chroma features and musical harmony are two entirely different concepts.

In this chapter, we introduce a set of low-level and mid-level features to account for the harmonic, timbre and tempo progression in music recordings. The chapter is structured as follows: Section 2.1 introduces basic notations and concepts from signal processing, Section 2.2 describes the extraction of *chroma* features which are related to the harmonic content of an audio signal, Section 2.3 explains the extraction of *Mel frequency cepstral coefficients (MFCC)* features which are used to model timbre, and Section 2.4 introduces *cyclic tempogram* features which account for local tempo changes in music recordings.

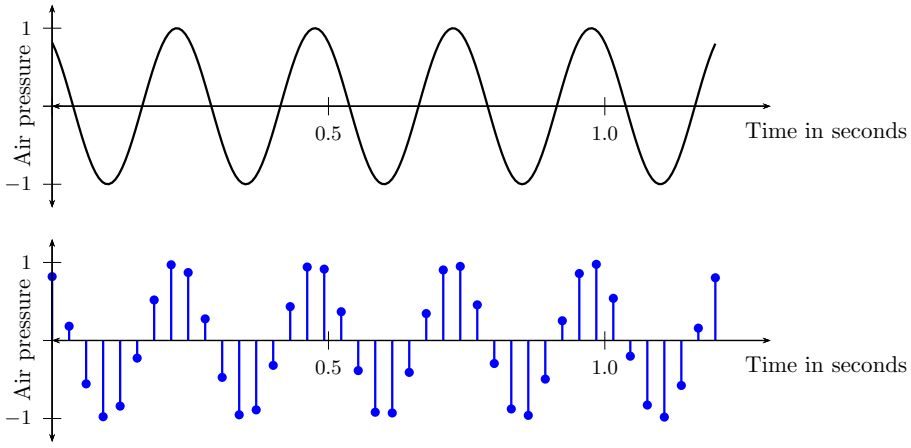


Figure 2.1. Top: CT signal (reproduced from [19]), **Bottom:** DT signal

2.1 Basics from Audio Signal Processing and General Remarks

As already stated before, a *sound signal* describes the variation of air pressure over time. Mathematically, this is a function $f : \mathbb{R} \to \mathbb{R}$ which maps a point in time $t \in \mathbb{R}$ to a sound-pressure $f(t)$. The function f describes a *continuous time (CT) signal*, see Figure 2.1 (top). As we deal with real-world signals which have a finite duration D, we restrict the domain of f to the interval $[0, D) \subset \mathbb{R}$. We assume f to be zero outside this interval, i.e. $f(t) = 0, \forall t \in \mathbb{R} \setminus [0, D)$.

The processing of real world signals in computers obliges us to discretize the CT signal. A *discrete time* (DT) signal can be computed from a CT signal by using an *equidistant sampling*

$$x(n) := f(T \cdot n) \tag{2.1}$$

with $T \in \mathbb{R}_{>0}$ and $n \in \mathbb{Z}$. The function x samples f at discrete points in time like shown in Figure 2.1 (Bottom). T is called *sampling period* and $1/T$ is the *sampling rate*. The DT signal x has again a finite duration as it is derived from f. The length of x is denoted by $N := \lfloor \frac{D}{T} \rfloor$ and the set of sample indices for x is then given by

$$[1 : N] = \{i \,|\, 1 \leq i \leq N \wedge i \in \mathbb{N}\} \tag{2.2}$$

We use the shorthand notation

$$x([a : b]) := \{x(i) \,|\, 1 \leq a \leq i \leq b \leq N \wedge i \in \mathbb{N}\} \text{ and } a, b \in \mathbb{N} \tag{2.3}$$

to index a fragment of x.

In the following, we will use f with a point in time t associated with CT signals, and x with sample index n associated with DT signals. The signal shown in Figure 2.1 was generated from a sine function and contains only one frequency. The waveforms we will deal with are usually more complex and contain not only many different frequencies, but also noise. An example is shown in 2.2.

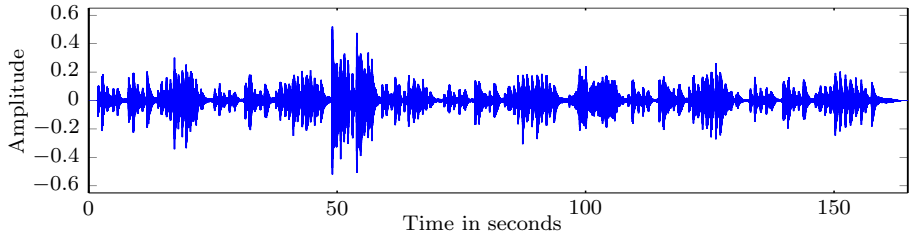


Figure 2.2. Waveform of recording of Chopin's Mazurka Op. 68 No. 3, sample rate 22050 Hz

The raise of digital recording techniques has lead to the situation that nearly all available music is already digitized. But in the case of old gramophone recordings we would still need to play back the recording and sample it at discrete time-points with digital recording equipment. The music recordings we use in our experiments are sampled at 22050 Hz.

Technically, a feature extraction method transforms an audio signal to a feature sequence $\mathbf{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N)$ with $\boldsymbol{v}_i \in \mathcal{F}$, $i \in [1 : N]$. In our case $\mathcal{F} = \mathbb{R}^d$ with $d \in \mathbb{N}$. To extract feature sequences, we need to consider information in the signal over a certain time-period. To do so, we use a window function $w : \mathbb{R} \to \mathbb{R}$ which follows a function g within the time interval $I = [-\frac{t_w}{2}, \frac{t_w}{2}]$ where $t_w \in \mathbb{R}$ denotes the *window size*. Outside the interval I, w is defined to be 0. In the discrete case, we denote the window size by N_w and the window is defined over the index set $[-\lfloor \frac{N_w}{2} \rfloor : \lfloor \frac{N_w}{2} \rfloor]$.

For the sake of clarity, we will use a constant window size in the feature extraction procedures. This results in several drawbacks. Firstly, the extracted features will be more noisy [28]. And secondly, passages in music recordings which are played in different tempi are treated the same way. The musical notes in passages of different tempi differ significantly in their duration. Consider two music recordings A1 and A2 which contain the same sequence of single musical notes, but in different tempi. In A1, every note has a duration of one second whereas in A2 every note lasts only for half a second. Now suppose that we extract a feature sequence on these recordings using a window length of one second with no overlap. Every feature vector corresponding to A1 would contain the information of one single note, whereas every feature vector corresponding to A2 would contain the information of two notes. In such a case, it would be beneficial to adapt the window length to the tempo in a music recording, such that recordings which contain the same musical material result in the same feature sequence. We will discuss the impact of the fixed window on the approach taken in this thesis in Section 6.1.

Generally, we extract features, such that the first feature is centered around $0s$. This is done by zero-padding the signal with respect to $\frac{t_w}{2}$. Adjacent windows overlap with a duration of t_{ov}. The time period which lies between the center points of two neighboring windows is called *hop size* $t_{hop} = t_w - t_{ov}$. The number of features per second is called *feature rate* and in the CT case, it is simply defined by $fr = \frac{1}{t_{hop}}$. The DT feature rate additionally needs to take the sampling rate into account, which leads us to $fr = \frac{1}{T \cdot N_{hop}}$, where N_{hop} is the corresponding DT hop size in samples.

In most of the

Many audio features need the frequency content of the signal for further processing. The time-domain representation of the signal can be transformed into a frequency representation by either applying the *Fourier transform* or *band-pass filters*. The Fourier transform of a CT signal is given by

$$c_\omega = \int_{t\in\mathbb{R}} f(t)e^{-2\pi i\omega t}\mathrm{d}t,$$

where ω is the frequency and c_ω is the Fourier coefficient. The amplitude $|c_\omega|$ is indicating the contribution of the frequency ω to the the signal and is also referred to as the *magnitude spectrum*. Usually, one is interested in the frequency content of a signal at a certain point in time $t_0 \in \mathbb{R}$. To this end, we use the *short-time Fourier transform (STFT)*, which is a windowed version of the Fourier transform:

$$STFT(t_0,\omega) = \int_{t\in\mathbb{R}} f(t)w(t-t_0)e^{-2\pi i\omega t}\mathrm{d}t$$

A detailed introduction of the Fourier transform and band-pass filters can be found in [19].

2.2 Chroma Features

In this section, we introduce chroma features which are closely related to the musical aspect of harmony. Chroma features are perceptually grounded by the fact, that two frequencies sound similar if one is a multiple of the other. In musical terms, this is called the *octave equivalence*. Technically, the octave relation of two pitches with the fundamental frequencies f_1 and f_2 is modeled by the formula:

$$f_1 = 2^k f_2, \quad k \in \mathbb{Z}. \tag{2.4}$$

We will use this relation to extract *chroma features* which were first introduced in [9] as *pitch class profiles*. Chroma features fold the pitch space into a 12 dimensional vector where each dimension corresponds to a chromatic pitch class. The chroma extraction basically consists of two steps:

1. Some pitch or frequency representation is computed.
2. The pitch/frequency representation is mapped to a 12-dimensional vector, representing the 12 chromatic pitch classes {C, C$^\sharp$, D, D$^\sharp$, ..., B} of the twelve-tone equal tempered tuning, the primarily used tuning system in western tonal music.

In step 1 we use *short-time mean-square power* (STMSP) features which are a pitch decomposition of the signal [19] : The signal is first decomposed into 88 pitch bands via a multi-rate filter bank. The pitch bands correspond to the pitches A0 to C8 (27.5-4185 Hz) which is the pitch range of a piano. The STMSP features are defined as follows:

$$\mathrm{STMSP(j,n)} = \sum_{k\in[n-\lfloor\frac{N_w}{2}\rfloor:n+\lfloor\frac{N_w}{2}\rfloor]} |x_j(k)|^2 \tag{2.5}$$

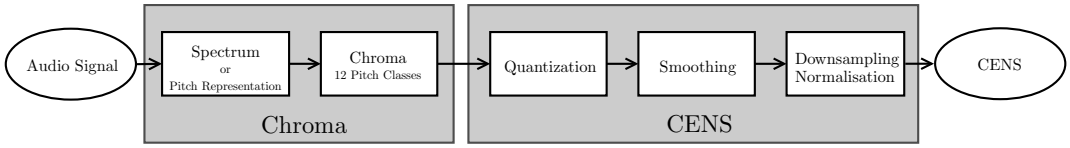


Figure 2.3. Chroma and CENS feature extraction pipeline

where x_j denotes a sub-band with $j \in [1 : 88]$ N_w denotes a fixed length window size. STMSP feature measure the local energy in each of the pitch bands.

2.2.1 CENS Features

In this section, we describe of *chroma energy normalized statistics* (CENS) which are an enhancement of the basic chroma features. Their extraction is basically a post-processing of the chroma features introduced above. First, we replace each chroma vector $\boldsymbol{v} = (v(1), \ldots, v(12))^\top$ by its ℓ^1 normalized version $v/\|v\|$. This makes the chroma more robust to differences in the sound intensities. Then a quantization of the energy distribution over the chroma bands is applied. The quantization is designed logarithmic way which was inspired by the logarithmic sensation of the sound intensity in the human ear. Afterwards, the chroma vectors are smoothed over N_s consecutive vectors to absorb local fluctuations within a chroma bin. Finally, the vectors are down sampled by a factor d and ℓ^2-normalised. The pipeline of the CENS feature extraction is summarized in Figure 2.3. Figure 2.4 shows an excerpt of a STMSP and a CENS sequence extracted from Chopin's Mazurka Op. 68 No. 3. We use the feature extractors from the Chroma Toolbox [20].

2.3 Mel frequency cepstral coefficients (MFCC)

Mel frequency coefficients (MFCC) are features originally developed for speech recognition [17]. In speech recognition, one is interested in detecting the words spoken in an audio signal. Words are composed of *phonemes*, and one can assume that they are pitch invariant, such that a speaker can produce the same phonemes with different pitches. This assumption is not entirely true, as phonemes can be described by their characteristic frequencies[1]. However, this loss of correctness is acceptable, as it allows us to draw a clearer picture of the construction and design of MFCCs. In a simplified modeling, the generation of voice in the human body can be modeled as originating from a sound source (vocal chords) and a sound filter (mouth). MFCCs basically separate the sound source from the filter. What remains in the end, are coefficients corresponding to the filter applied to the sound source.

For music, we can use the same simplified modeling. The sound of a musical instrument is produced by its sound source, e.g. a piano string, and a filter, e.g. the sounding body of a piano, which modifies the signal of the sound source. After separating the sound source from the filter, the filter response should give a good description of the musical timbre in

[1]Those frequencies are called *formants*. The interested reader can find an in-depth explanation in [18]

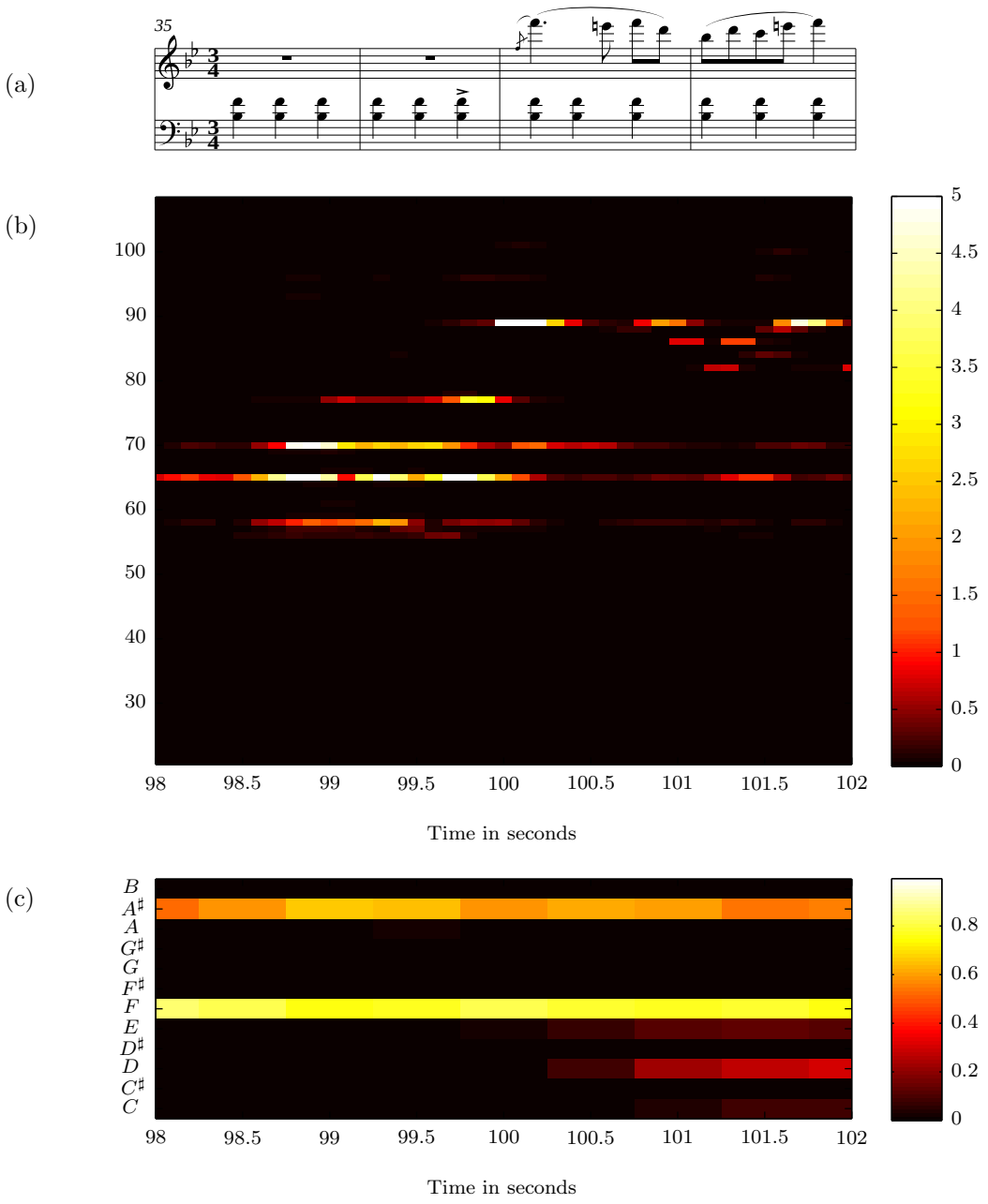


Figure 2.4. Features of Chopin Mazurka Op. 68 No. 3 played by Cortot in 1951
(a): Excerpt from musical score corresponding to (b) and (c), measure 35-38 (Poco più vivo)
(b): STMSP feature sequence on measure 35-38 **(c):** CENS feature sequence on on measure 35-38

the sound signal. Again, this is a strongly simplified explanation of how a musical sound is created.

MFCCs are computed from a spectral representation. The frequencies are transformed to the Mel scale by applying a triangular filter bank. The Mel scale models the non-linearity of human pitch perception, see Figure 2.6. The Mel frequency mapping is defined as

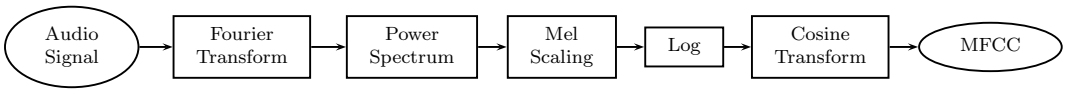


Figure 2.5. MFCC feature extraction pipeline

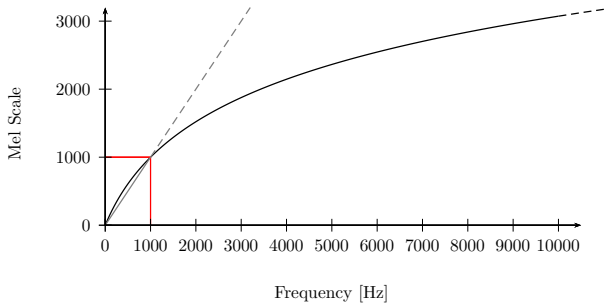


Figure 2.6. Frequency plotted against Mel scale. Black Line: $M(f)$, Grey Line: linear frequency

follows:

$$M(f) = 2595 \log_{10}\left(\frac{f}{700} + 1\right)$$

where f denotes the frequency in Hz.

After mapping the signal to the Mel Scale, the Logarithm is applied to the Mel Frequencies to account for the logarithmic sensation of sound pressure. Until now, the the spectral coefficients are still highly correlated. To decorrelate the feature vectors, the *Discrete Cosine Transform (DCT)* is applied in the last step. The DCT is a transformation similar to the Fourier transform, but with real valued output. The first MFCC dimension is the loudness, which follows from the definition of the DCT. In practice, the first coefficient can be discarded. We will actually discard the first three, as the first dimensions of the MFCCs are still correlated to the loudness.

We additionally apply the smoothing and downsampling explained in the CENS feature extraction in Section 2.2.1. A summary of the MFCC feature extraction procedure is given in Figure 2.5. Figure 2.7 shows an example of a 40 dimensional MFCC feature sequence on Beethoven's Fifth. Figure 2.7a shows the raw MFCC feature sequence whereas in Figure 2.7b we applied an ℓ^2 normalization. The framed region in the figure marks the oboe solo. At this position in the piece, one would expect a timbre change. This timbre change is reflected in the MFCCs, as the feature sequence corresponding to the solo stands out clearly.

In Figure 2.7b, we can clearly distinguish the region of the oboe solo whereas in Figure 2.7a, the loudness is dominating the feature sequence and the visualization does not yield any information apart from the loudness dimension. Usually, the first coefficients in a MFCC feature vector are discarded, as they are highly correlated to loudness. There are a lot of different opinions which MFCC dimensions should be discarded [1]. We keep the

$4^{th} - 13^{th}$ dimensions of the MFCC. Our feature extractor is based on [23].

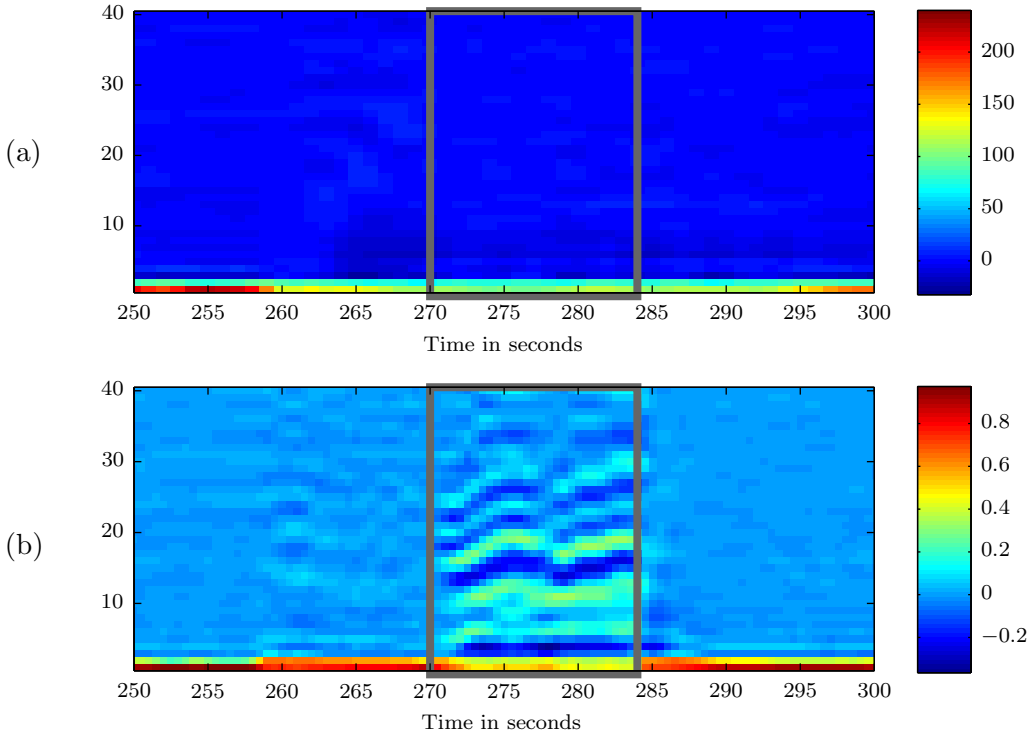


Figure 2.7. MFCC feature sequences on excerpt of Beethoven's Fifth (Op. 67 No. 01) just before and after the oboe solo. The region which is marked by the gray box corresponds to the oboe solo. **(a):** Raw MFCCs, note that the first dimension is the loudness which dominates the feature sequence. **(b):** ℓ^2-normalized MFCCs.

2.4 Tempogram Features

One other important aspect of music is the *tempo*. Loosely speaking, the tempo is the speed in which a piece of music is played. The musical tempo is measured in *beats per minute (BPM)* which indicates the number of pulses per minute. There are different levels of tempo, such as the measure, tatum, and tactus level [15]. The *tatum* refers to the shortest musical duration in a piece of music. The tactus is also referred as *beat* or foot-tapping rate which is the predominant perceived pulse. This level is very ambiguous. It is highly probable that a musically untrained person would not determine the beat on the right tempo level, whereas most musician do agree on the beat level.

The tempo in music recordings can change locally. These tempo changes are part of an expressive performance. One can find that at certain passages, musicians accelerate or slow down. This is done to emphasize a certain element in the music. Sometimes, these tempo changes do occur at structural boundaries, e.g. a slow down (ritardando) just before a section in the music with higher tempo. To determine those passages we do not need the exact musical beat. In this case it is sufficient to rely on a mid-level representation, which reveals the *local tempo changes*.

2.4.1 Cyclic Tempograms

For this purpose we will use *cyclic tempograms* [11], which are features conceptually similar to chroma features, but for the tempo domain. Recall the octave equivalence from Section 2.2. Similar to Equation (2.4), we define two tempi τ_1 and τ_2 to be octave equivalent if they fulfil the relation

$$\tau_1 = 2^k \tau_2, \quad k \in \mathbb{Z} \tag{2.6}$$

This results in tempo equivalence classes similar to the pitch classes from the chroma features. Cyclic tempograms are derived from tempograms, which are time-tempo representations, similar to chroma features which are derived from a time-pitch representation. A tempogram is computed from a *novelty curve* which indicates sudden changes in the signal. The novelty curve is extracted by exploiting the fact, that note onsets usually coincide with a sudden change in the signal's energy and the spectrogram [11], see Figure 2.8 (a). More precisely, one computes a short time Fourier transform X of the signal, applies a logarithmic compression on the magnitude spectrum $|X|$, which leads to the compressed spectrum Y. Then we compute the discrete derivative of Y, where we only keep the non-negative part to emphasize note onsets and to discard note offsets. Finally, we subtract the local average and only keep the non-negative part to construct final novelty curve Δ

To obtain a tempogram from the novelty curve Δ, we will use the Fourier transform to compute a *Fourier tempogram*, see Figure 2.8 (b), and the unbiased local autocorrelation to compute an *autocorrelation tempogram*, 2.8 (c). The tempograms are discussed in detail in [11]. The two tempograms are then folded into a cyclic tempogram by applying the tempo octave equivalence from Equation (2.6). We use the feature extractors provided in the Tempogram Toolbox [10].

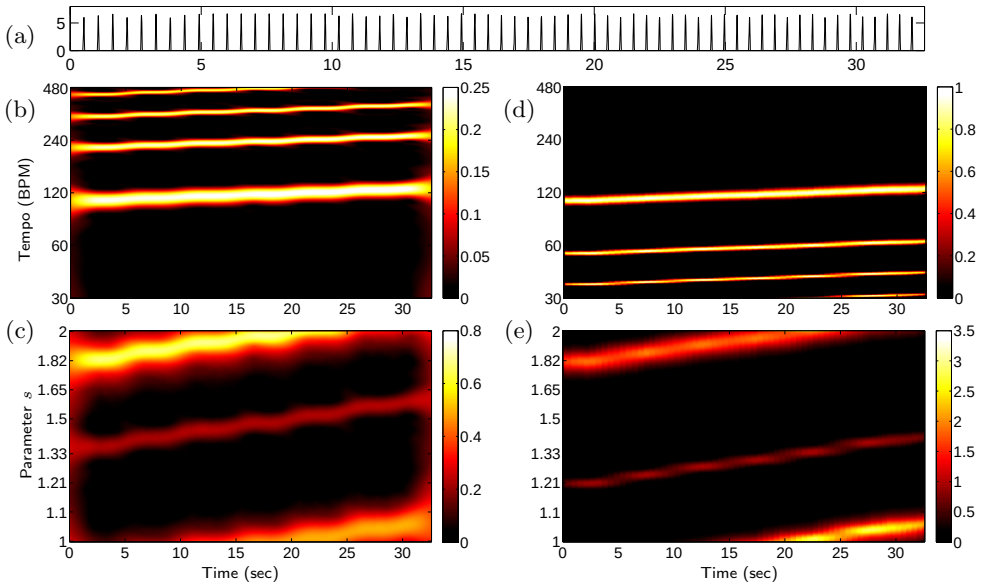


Figure 2.8. Tempo features for a click track with increasing click frequency: **(a)** onset signal for click track, **(b)** Fourier-based tempogram, **(c)** cyclic Fourier-based tempogram, **(d)** autocorrelation-based tempogram, **(e)** cyclic autocorrelation-based tempogram. Reproduced from [11].

2.5 Feature Settings

In this section, we fix the feature settings which we will keep constant throughout this thesis.

Our audio files are all have a sampling rate of 22050 Hz. We will extract all feature with a feature rate of 2 Hz. Concretely, the parameter settings for CENS and MFCC features are the following: We use a constant window size of $N_w = 4410$ samples with half overlapping windows (2205 samples). In this case, the hop size is equal to the window overlap. The smoothing of the feature vectors is set to $N_s = 21$ and a downsampling factor $d = 5$ is used.

The tempogram features are derived from a novelty curve which has a sample rate of 200 Hz. To extract a tempogram of 2 Hz from this novelty curve, we use a hop size of 100 on the novelty curve.

Contrary to popular belief, MIDI doesn't "sound bad" – it has no sound at all.

David Battino

3

Music Synchronization

Music can appear in many different representations. A piece of music can be represented in different ways, e.g. its score or by an audio recording. The score itself can be encoded in an image or in a digital music notation format such as *MusicXML*. For one piece of music, there can exist many different instances or versions in form of audio recordings or different representations. In the case of audio recordings, they are called *interpretations* of the piece. It is an important point to make the difference between the piece of music itself, and its different instances in form of interpretations and different music representations. In Music Synchronization, one is interested to find an alignment between two instances of a piece of music. The task can be to match the notes in the score to an audio recording of the piece. If the score is represented by an image, the problem is to match x- and y-coordinates of the image to a time point t in the audio recording. In our case, we already have a symbolic representation of the music in form of a MIDI file. MIDI (Musical Instrument Digital Interface) is a protocol which was originally developed to control digital instruments [19]. MIDI can be thought of as a representation between the musical score and an audio recording. The format allows to encode a certain degree of expressiveness. It is even possible to record a performance with a digital player piano to a MIDI file and then play it back again on the piano. On the other hand, MIDI is no replacement for a musical score, because the latter contains additional performance instructions and the typesetting of a musical score is an art in itself. The most important part of MIDI files for us are the *note onset* and *note offset* events which determine the start and end time of a musical note. Figure 3.1 shows a musical score excerpt together with a piano roll visualization of a MIDI version and a waveform of an audio recording. To synchronize a MIDI version of a piece with an audio version, we need to transform both versions into a suitable sequence of features. Those sequences can then be aligned using a technique called *dynamic time warping*, which we will introduce in 3.2.

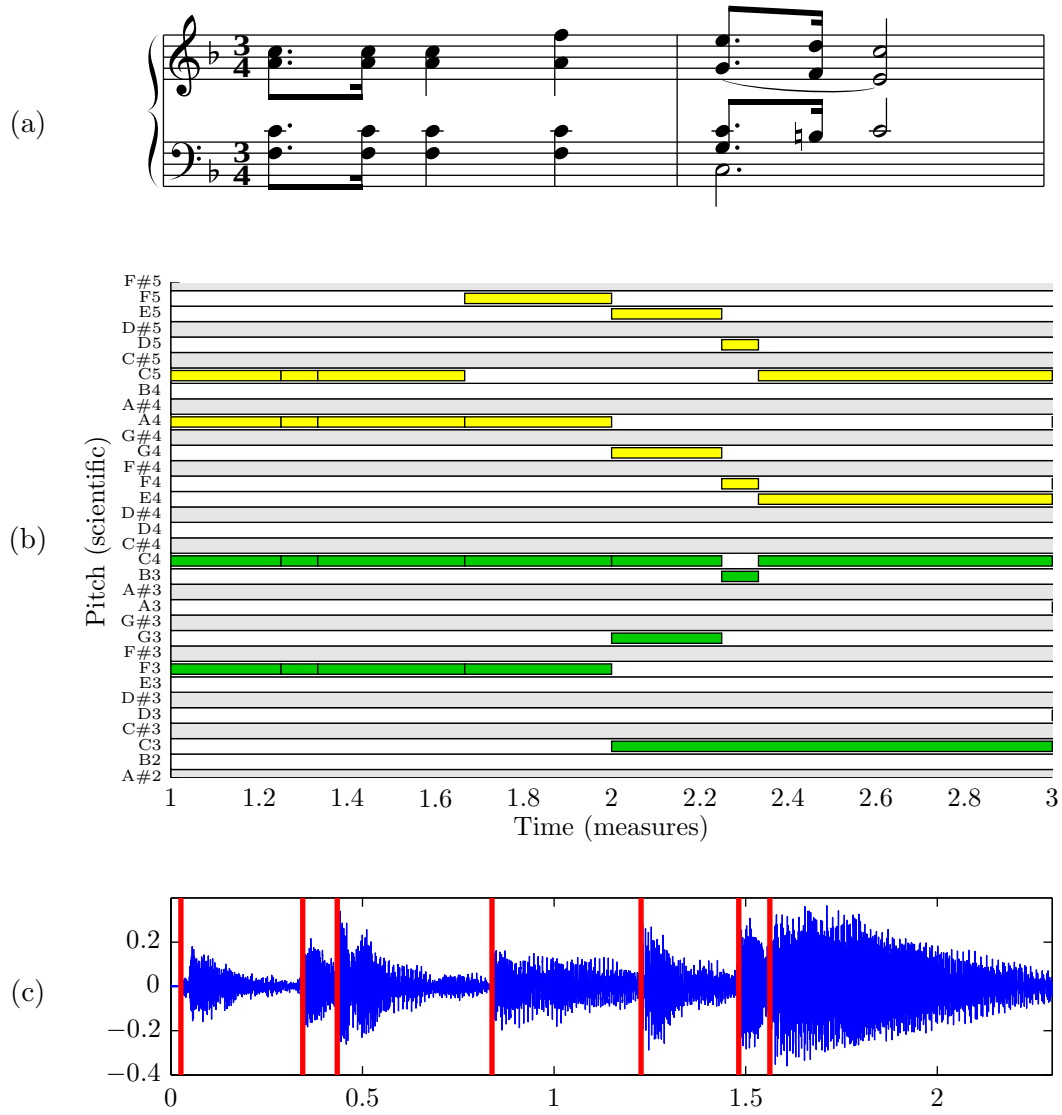


Figure 3.1. **(a)**: Musical score. **(b)**: Piano roll visualization of MIDI version of Mazurka Op. 68 No. 3. Horizontal lines indicate the duration of notes and their vertical position is the pitch height. **(c)**: Waveform. The vertical lines indicate the note onsets.

3.1 Related Work

Music synchronization using dynamic time warping has been first introduced by Orio and Schwarz [22]. They used spectral features to model pitch, note attacks and silence. Dannenberg et al. introduced the use of chroma features in dynamic time warping based music synchronization. Recently, the music synchronization has been enhanced with the introduction of DLNCO (decaying locally adaptive chroma onset) features, which is a hybrid of onset and chroma features [5, 4]. DLNCO features are derived from STMSP features. First, the discrete temporal derivative of an STMSP feature sequence is calculated and half-wave rectified, which results in keeping only the part of the derivative responsible for note onsets and discarding note offsets. This sequence is then transformed into a chroma vector by adding up a logarithmically scaled versions of each band to account for the logarithmic sensation of sound intensity. These chroma vectors are then normalized and divided by the the local maxima of the norms in a neighbourhood of 2 seconds (one second to the left and one second to the right). This accounts for the fact that low onsets within loud passages are usually less important than in quiet passages. The resulting features are called LNCO (locally adaptive normalize chroma onset) features. To finally obtain DLNCO features, each feature is copied n times and the copies are multiplied by decreasing weights. This results in features starting with the original onsets of the LNCO features followed by successively decaying versions of it. We will use the DLNCO features in the synchronization pipeline of our experiments and outline the basics of dynamic time warping and music synchronization in the following sections.

3.2 Dynamic Time Warping (DTW)

Dynamic time warping (DTW) is a technique to find an optimal alignment between two given (time-dependent) sequences under certain restrictions [19]. Our goal is to align the sequences $X = (x_1, \ldots, x_N)$ and $Y = (y_1, \ldots, y_M)$ with $N, M \in \mathbb{N}$. X and Y can be a discrete signal or any feature sequence. In our case, X and Y are ordered subsets of a *feature space* $\mathcal{F}$. To compare the two sequences, we need a *local cost measures*, which is a function $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_{\geq 0}$ whose purpose is to measures the similarity or dissimilarity between two feature points $x_n \in X$ and $y_m \in Y$, with $n \in [1 : N]$ and $m \in [1 : M]$. Optimally, two similar features yield to a low cost and vice versa. This is why c is also called *distance measure*. The matrix

$$C := (\ c(x_n, y_m)\ )_{n \in [1:N], m \in [1:M]}$$

which contains all pairwise costs of the pairs $(x_n, y_m) \in X \times Y$ is called *cost matrix*. Figure 3.2 (Left) shows an example of C.

Intuitively, the alignment of X and Y follows would follows a valley of low cost in C. We now formalize the notion of an alignment, which we call *warping path*. A warping path is a sequence $p = p_1, \ldots, p_L$, with $p_l = (n_l, m_l) \in [1 : N] \times [1 : M]$ and $l \in [1 : L]$ satisfying:

1. Boundary condition: $p_1 = (1, 1), p_L = (N, M)$
2. Monotonicity condition: $n_1 \leq n_2 \leq \ldots \leq n_L, \quad m_1 \leq m_2 \leq \ldots \leq m_L$

3. Step size condition: $p_{l+1} - p_l \in \{(1,0),(0,1),(1,1)\}$

The boundary condition ensures that the beginning (ending) of the first sequence and the beginning (ending) of the second sequence are matched. The step size condition guarantees that every element of the first sequence is matched with an element in the second sequence. It also ensures that the time-ordering of the sequences is respected. The monotonicity condition redundant as it is implied by the step size condition, but we nevertheless state it here for the sake of clarity. Note, that one element in a sequence can be matched to several elements in the other sequence. In the case of music recordings, this happens if the sequence X is played in a shorter time than in sequence Y, which yields to a mapping $\{\ldots,(n,m),(n,m+1),(n,m+2),\ldots\}$. As we compute alignments between feature sequences and a feature always correspond to a time range, it is important to use a meaningful interpolation function if the same feature is mapped to several features. We will address this problem in Section 3.4. The *total cost* c_p of a warping path is defined as

$$c_p(X,Y) := \sum_{l=1}^{L} c(x_{n_l}, y_{m_l}).$$

The optimal warping path p^* is the warping path which has minimal costs. The *DTW distance* $\mathrm{DTW}(X,Y)$ is defined as $\mathrm{DTW}(X,Y) := c_{p^*}(X,Y)$. The algorithm to find p^* is based on *dynamic programming* which divides a problem into smaller sub problems. First we need to define the *accumulated cost matrix* $D(n,m) := \mathrm{DTW}(X(1:n), Y(1:m))$ with $n \in [1:n]$ and $m \in [1:m]$. $X(1:n)$ and Y(1:m) denote prefix sequence of X and Y.

The optimal warping path p^* is computed in reverse order of the indices by backtracking over the matrix D, starting at $p_L^* = (N,M)$. Note that the length L of p^* is constrained by $\max(N,M) \leq L \leq N+M-1$, which follows from the step size and boundary condition. Suppose $p_L^*,\ldots,p_l^*$ have been computed, $1 < l \leq L$ and let $p_l^* = (n_l, m_l) = (n,m)$, then the formula to compute the remaining warping path is:

$$p_{l-1} = \begin{cases} (1, m-1), & \text{if } n = 1 \\ (n-1, 1), & \text{if } m = 1 \\ \underset{(i,j)\in I}{\operatorname{argmin}} D(i,j), & \text{if } n > 1, m > 1 \end{cases} \tag{3.1}$$

$$\text{where } I := \{(n-1,m-1),(n-1,m),(n,m-1)\}.$$

Figure 3.2 (Right) shows an example of D with the corresponding warping path.

This is the basis for dynamic time warping. There exist many refinement strategies such as different step size conditions or additional local weights for the accumulated cost matrix D

$$D(n,m) = \min \begin{cases} D(n-1,m-1) + w_d \cdot c(x_n, y_m) \\ D(n-1,m) + w_h \cdot c(x_n, y_m) \\ D(n,m-1) + w_v \cdot c(x_n, y_m) \end{cases} \tag{3.2}$$

to enforce a diagonal direction of the warping path. A good choice to enforce a diagonal direction is $(w_d, w_h, w_v) = (2, 1.5, 1.5)$. Furthermore one can coarsen the feature sequence

	Index	1	2	3	4	5	6	7
Features	A	a	b	c	d	d		
	B	a	a	a	a	b	c	d

Warping Path	A	1	1	1	1	2	3	4	5
	B	1	2	3	4	5	6	7	7

Table 3.1. Skteched warping path for two feature sequences.

and put global constraints on the warping path, such that only a part of D around the main diagonal has to be considered, which significantly reduces the computational complexity. The latter two refinements yield to the multi-scale DTW approach, where a dynamic time warping is performed on a coarse resolution of D, which then constrains the computation of the warping path on a finer grained levels. A detailed discussion can be found in [19].

3.3 Time Assignment Function

Recall that a warping path assigns elements of one feature sequence to another feature sequence. As a feature is associated with the time range determined by the feature resolution, we can interpret a warping path as an assignment of time ranges. To calculate the corresponding point in time in one music recording from an arbitrary point in time in another music recording, we need to apply interpolation techniques on the warping path. As time is a strictly monotonically increasing phenomenon (at least in classical physics). Thus, a desirable property of the interpolation technique would be that a set of strictly monotonically increasing points in time in one recording is assigned to a strictly monotonically increasing set of points in time in the other recording. Previous approaches relied on staircase interpolations which are only monotonically increasing and may result in assigning a set of points in time in one recording to one single point in time in another recording. Furthermore, this kind of simplified modeling leads to abrupt directional changes and strong local temporal distortions [4]. In this thesis, we use an interpolation technique introduced in [4] which leads to a strictly monotonically increasing *time assignment function*. The basic idea is to compute local distortion factors from the time range assignment defined by the warping path. Consider the warping path from Table 3.1 and assume a feature rate of 50 Hz such that each feature refers to a time range of $20ms$. The feature at index 1 from sequence A is assigned to the features $1, 2, 3, 4$ in sequence B. This means that the time range of $20ms$ corresponding feature 1 in sequence A is mapped to a time range of $80ms$ in sequence B. The local distortion distortion factor in this case would be $20ms/80ms = 1/4$. Technically, the local distortion factors are then encoded in a density function. The final time assignment function is obtained by integrating over the density function [4]. We refer to the resulting function as

$$timewarp(t, p, sr_p) \tag{3.3}$$

where t is a point in time, p is the warping path and sr_p is the sample rate of the warping path. For a detail derivation of the time assignment function, we refer to [4, 3].

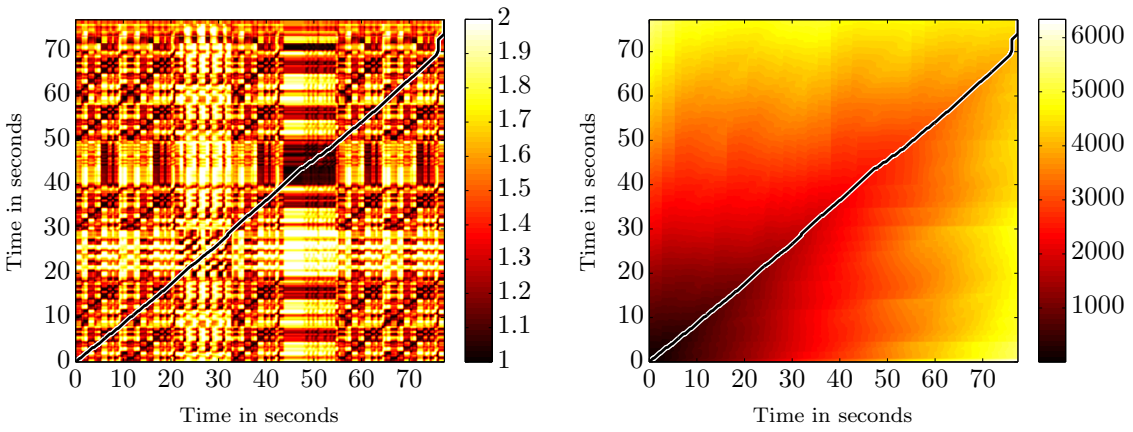


Figure 3.2. Left: Visualization of warping path on cost matrix C, computed on Mazurka Op. 68 No. 3 played by Ashkenazy in 1981 and a MIDI version of the piece. **Rigth:** Accumulated cost matrix D with warping path.

3.4 Synchronization Pipeline

In the previous sections, we introduced the basics of music synchronization. The synchronization technique we apply in this thesis incorporates several enhancements. First, it combines two cost matrices from chroma and DLNCO features. This approach incorporates the temporal accuracy from onset features while preserving the robustness gained from chroma features. Secondly, we use a multi-scale implementation. Thirdly, we use a sophisticated interpolation function on the warping path which closely follows the original course of the warping path. A detailed discussion 3.1.

In general, we will compute warping paths from features that have a time resolution of $20ms$.

4

Novelty Detection

A piece of music is usually structured in different segments. The segments define regions which are restricted by two boundaries marking their start and end point. A boundary is the point in time of the transition between two segments. The goal of boundary detection is to find the transitions between consecutive musical segments. Two consecutive segments often contrast with respect to one ore more musical aspects. The point in time where some musical aspect changes is therefore a potential candidate of a structural boundary. The aim of novelty detection is to find the points in time in a music recording, where a musical aspect changes. There are many musical aspects which can change between two segments, such as harmony, melody, pitch, rhythm, tempo, loudness and timbre. We restrict us to the musical aspects harmony, rhythm and timbre for which we introduced audio features in Chapter 2. In this chapter, we explain how the audio features can be used in a novelty detection algorithm. The novelty-based structure detection which we introduce in this chapter aims to find boundaries between contrasting regions. In Section 4.1 we introduce the concept of a self distance matrix which is the same concept as the cost matrix introduced in Chapter 3. The algorithm for novelty detection is presented in Section 4.2.

4.1 Self Distance Matrix

As a music recording usually consists of different structural parts, which may repeat and differ under a certain aspect such as harmony, it is a promising strategy to analyze the self-similarity of the recording. The self-distance matrix (or self-similarity matrix) is a useful tool to analyze the pairwise similarity of all point in times in a music recording with respect to a certain feature and a similarity or distance measure. The use of a similarity measure leads to a *self similarity matrix (SSM)* and the use of a distance measure leads to a *self distance matrix (SDM)*. A distance measure is a function $d : \mathcal{F} \to \mathbb{R}$ which

expresses how distinct two feature vectors $\boldsymbol{v}_i, \boldsymbol{v}_j \in \mathcal{F}$ are, whereas a similarity measure $s : \mathcal{F} \to \mathbb{R}$ describes how similar they are. To compute the self distance matrix, we adapt the similarity measure s_e introduced in [2] which combines the cosine similarity and an exponential function:

$$s_{cos}(\boldsymbol{v}_i, \boldsymbol{v}_j) := \frac{\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle}{\parallel \boldsymbol{v}_i \parallel \parallel \boldsymbol{v}_j \parallel} \tag{4.1}$$

$$s_e(\boldsymbol{v}_i, \boldsymbol{v}_j) := \exp\left(s_{cos} - 1\right) \quad . \tag{4.2}$$

Note that s_{cos} is normalized and maps to $[-1, 1]$. Using a distance based on the cosine distance leads basically to Euclidean normalized vectors. This may not be the ideal choice for every feature and task as the chroma space is not Euclidean. Also the distance measure might not be optimal. Consider the three chroma vectors:

$$\begin{aligned}
\boldsymbol{v}_1 &= [1,0,0,0,0,0,0,0,0,0,0,0]^\top, \\
\boldsymbol{v}_2 &= [0,1,0,0,0,0,0,0,0,0,0,0]^\top, \\
\boldsymbol{v}_3 &= [0,0,0,0,0,0,0,1,0,0,0,0]^\top.
\end{aligned}$$

They all share the same pairwise Euclidean distance and cosine distance of zero. On the other hand, the musical pitch interval from C to to $D^\sharp$ is a minor second, whereas from C to G it is a fifth. We are aware of this discrepancy but will nevertheless use a distance measure based on the cosine distance, as it is not the focus of this work to find the optimal distance measure.

From now on, we use SDMs. Thus we need to transform the similarity measure into a distance measure. Due to the normalized features, this can be easily done by taking the difference $d(\boldsymbol{v}_i, \boldsymbol{v}_j) = 1 - s(\boldsymbol{v}_i, \boldsymbol{v}_j)$. We further modified the similarity measure from [2] by adding the parameters α and β:

$$d_e(\boldsymbol{v}_i, \boldsymbol{v}_j) := \left(1 - \exp\left(\alpha \left(\frac{\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle}{\parallel \boldsymbol{v}_i \parallel \parallel \boldsymbol{v}_j \parallel} - 1\right)\right)\right)^\beta \quad . \tag{4.3}$$

We call α and β homogeneity parameters, where $\alpha \geq 1$ takes effect on the homogeneity of dissimilarity values (i.e. high values) and $\beta \geq 1$ affects the homogeneity of similarity values (i.e. low values). If not stated otherwise, we will use the standard parameters $\alpha = \beta = 1$ which reduces Equation (4.3) to $1 - s_e$. The SDM on a feature sequence of length N is then defined as

$$\mathcal{D}(i, j) := d_e(\boldsymbol{v}_i, \boldsymbol{v}_j) \text{ for } i, j \in [1 : N]. \tag{4.4}$$

If the distance measure is symmetric, it directly follows that $\mathcal{D}(i, j) = \mathcal{D}(j, i)$. Therefore, we only need to compute the main diagonal and the upper part of the values to fill the entire matrix. This reduces the number of computations from N^2 to $\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$. The SDM is zero valued on the main diagonal as a feature compared to itself has distance zero. Figure 4.6 (b) shows a SDM on a music recording. Diagonal stripes off the main diagonal indicate similarities between segments. Stripes with a slope different to 1 indicate

a difference in tempo. If two consecutive homogeneous segments in a music recording are different to each other in relation to a certain feature, the SDM consists of low values for intra-segment comparisons and high values for inter-segment comparisons. Ideally, this leads to a checkerboard pattern. An example of SDMs along with an idealized checkerboard pattern is shown in Figure 4.1. One of the reasons why the computed matrices in the figure

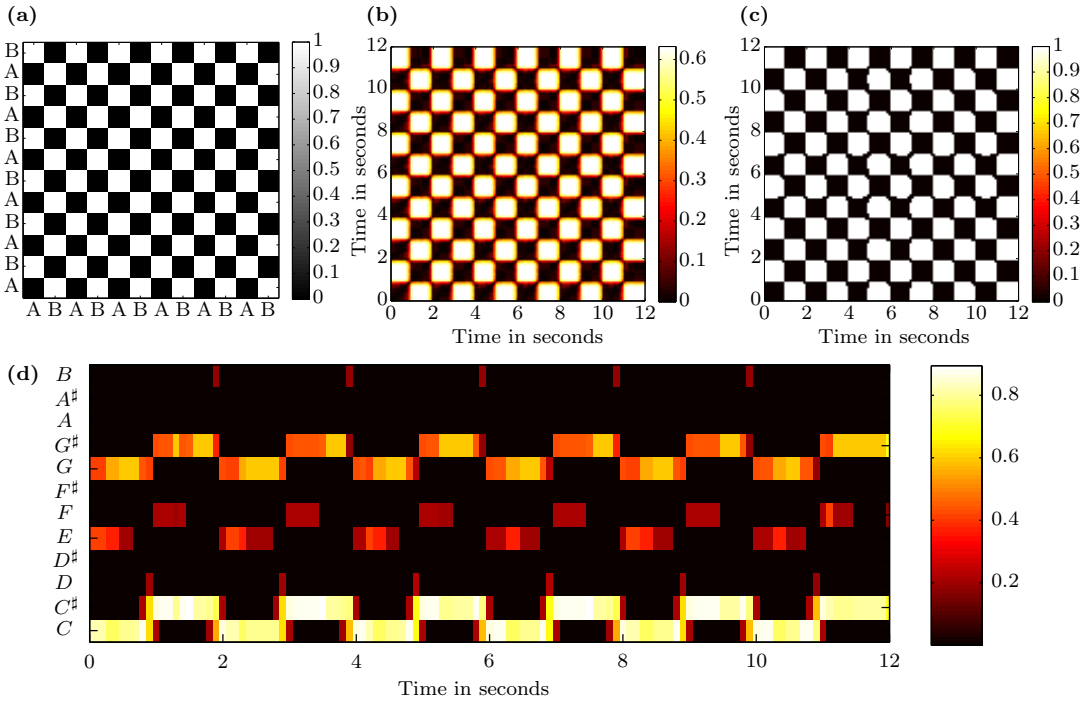


Figure 4.1. **(a)**: Ideal checkerboard pattern of a piece with two structures A and B, **(b)**: SDM with $\alpha = \beta = 1$, computed from (d). **(c)**: SDM with $\alpha = 15$, $\beta = 10$ on the same feature sequence as (b). A threshold was applied to show that we can nearly compute the ideal pattern as shown in (a). **(d)**: CENS feature sequence of a recording which consists of the repetition of the two alternating pitches C and $C^\sharp$.

show artifacts in their checkerboard patterns is the fixed length window size which does not incorporate duration information for individual notes. Additionally, the note onsets from percussive instruments (e.g. a piano) make the chroma vectors more noisy. This is due to the fact, that a percussive onset has nearly an infinite slope. This results in a spectral vector at the point in time of the onset whose energy is smeared over all frequencies bins. This point becomes clear, if one looks at an impulse signal $\delta(t)$ which is ∞ at $\delta(0)$ and zero everywhere else. An impulse can be interpreted an the extreme version of an onset. The Fourier transform of the impulse results in $c_\omega = \int_{t\in\mathbb{R}} \delta(t) e^{-2\pi i \omega t} \mathrm{d}t = 1 \, \forall \omega \in \mathbb{R}$, In the following we will explain how the checkerboard pattern in the SDM can be used to detect the boundaries between contrasting regions in music recordings.

4.2 Correlation based Novelty Extraction

In this section we will explain a method originally introduced by J. Foote in [6]. The basic idea of this algorithm is to exploit the checkerboard like pattern which arises in the matrix $\mathcal{D}$ around boundaries between two consecutive homogeneous feature sequences. Correlating the diagonal of $\mathcal{D}$ with such a pattern generates a *novelty curve* $\mathcal{N}$ which we expect to have high values at segment boundaries and low values for intra-segment regions. A simple 5×5 checkerboard pattern is given by the kernel matrix

$$\mathcal{K}_{BC} = \begin{bmatrix} -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 & -1 \\ 1 & 1 & 0 & -1 & -1 \end{bmatrix}$$

where BC stands for box checkerboard. The matrix $\mathcal{K}_{BC}$ consists of 4 equally sized blocks. Generally, a checkerboard kernel is a matrix $\mathcal{K} \in \mathbb{R}^{(2B+1)\times(2B+1)}$ where $L = (2B + 1)$ denotes the kernel length and B is the length of each of the 4 blocks in $\mathcal{K}$. The novelty curve $\mathcal{N}$ is then computed with the following formula:

$$\mathcal{N}(i) = \sum_{k=-B}^{B} \sum_{l=-B}^{B} \mathcal{K}(k,l)\mathcal{D}(i+k, i+l) \quad . \tag{4.5}$$

To avoid undefined novelty values at the first and last index $i \in \{1, N\}$, the matrix $\mathcal{D}$ is zero padded with respect to the block length B. As this could lead to negative values in the novelty curve, we only keep the non-negative part by applying the max function:

$$\mathcal{N}(i) = \max(0, \mathcal{N}(i)). \tag{4.6}$$

Note that both $\mathcal{D}$ and $\mathcal{K}$ are symmetric. Thus, we only need to compute half of the values in Equation (4.5), namely the ones fulfilling $k \geq l$. Additionally, we only need to compute a small stripe of the $\mathcal{D}$, i.e. $\mathcal{D}(i, i-j)$ fulfilling $0 \leq i - j \leq L$ with $i, j \in [1 : N]$. This reduces the complexity of the algorithm to $\mathcal{O}(N \cdot L^2)$.

4.3 Checkerboard Kernels

4.3.1 Even versus Odd Kernel Length

Above we defined the matrix $\mathcal{K}$ with an odd length $L = (2B + 1)$. The reason why we favor the odd kernel over an even kernel length $L = 2B$ is the following. Remember that we extract feature sequences, such that the first feature always corresponds to $0s$. For every remaining feature, its point in time is located in the center of the window which was used in its computation. Suppose we use a feature rate of 1 Hz, then the second feature is located at $1s$. Consider an even kernel with $L = 4$ which would have its center at the undefined index 2.5. If we use such a kernel on a SDM, we would actually compute the novelty between two time frames, and hence, introduce a half frame shift. The first novelty

point would be located at 0.5 seconds which is between the first and second feature. By using an odd kernel, we can simply map the novelty value at index i to the feature with the same index. Figure 4.2 shows two novelty curves computed with an odd and an even kernel on a synthetic SDM. Note that the novelty curve computed with the even kernel shows a sharper peak but the corresponding point in time is located between two features. The novelty curve extracted with the odd kernel has its novelty values located at the points in time corresponding to the center points of the features, but its peak is smeared over two locations.

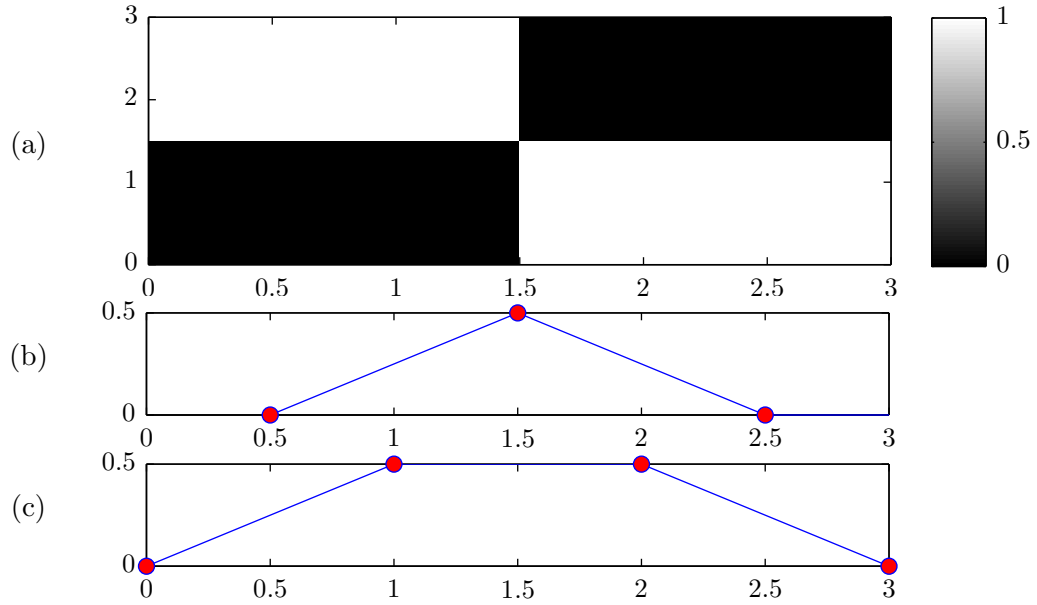


Figure 4.2. Example of novelty curves resulting from an even kernel length compared to an odd kernel length. The data is synthetic and we assume a feature resolution of 1 Hz, such that the SDM matrix has one value at each integer pair (x,y). The red points in the novelty curves (b) and (c) mark the computed values. The blue lines are linear interpolations between the computed values. **(a):** Synthetic SDM which would correspond to a feature sequence of length 3. **(b):** Novelty curve resulting from an even kernel length $L = 2$. The peak is located between at 1.5 which is located between two features. **(c):** Novelty curve resulting from an odd kernel with $L = 3$. The peak is not as sharp as in (b), but each novelty value can clearly be assigned to a feature index.

4.3.2 Box Checkerboard Kernel

An arbitrary box checkerboard kernel can be constructed with

$$\mathcal{K}_{BC}(k, l) = \text{sgn}(k) \cdot \text{sgn}(l) \tag{4.7}$$

where $k, l \in [-B : B]$ and $\text{sgn}(\cdot)$ is the sign function.

4.3.3 Gaussian Checkerboard Kernel

Generally, the contrast of two objects is defined by their difference under a certain attribute. More precise, these attributes need to show a high difference in a local neigh-

borhood, namely around the objects boundaries. We apply this concept to a piece of music where the attributes which result in a contrast are harmony, tempo, loudness and timbre. To integrate the importance of the local neighborhood in the kernel, we weight the BC kernel by a 2D Gaussian. Thus, in the novelty detection, the musical events in the immediate proximity of a particular point in time receive a higher weighting and the ones which are temporally further away receive a lower weighting. Generally, the 2D Gaussian checkerboard (GC) kernel is constructed as follows:

$$\mathcal{K}_{GC}(x, y, \sigma) = \mathcal{K}_{BC}(x, y) \cdot \exp\left(-\frac{1}{2}\left(\frac{k^2 + l^2}{(\sigma B))^2}\right)\right) \ , \ k, l \in [-B : B] \tag{4.8}$$

where σ is the standard deviation of the Gaussian. Note that the shape of the kernel is invariant to the kernel length L due to the normalization with the block length B in the Gaussian. Figure 4.3 shows two GC kernels of different length.

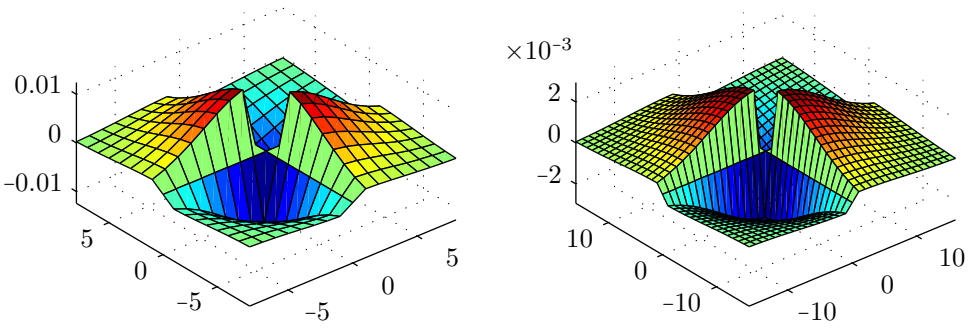


Figure 4.3. Two Gaussian checkerboard kernels of different length with $\sigma = 0.5$ **(Left):** $L = 17$. **(Right):** $L = 33$.

4.3.4 Kernel Normalization

Depending on the kernel type and its length, the resulting novelty curve will always have a different range. To make the response of the kernel invariant to its length, we apply a normalization factor to the kernel

In the case of a box checkerboard kernel, it is easy to see that the maximum possible value in a novelty curve is $(L-1)^2$ if the values of the SDM lie in the interval $[0, 1]$. Thus, its normalization factor is: $\frac{1}{(L-1)^2}$. In the case of a Gaussian kernel, the maximum novelty value depends on the length of the kernel L and the standard deviation σ. According to Equation (4.8), the theoretical normalization factor would be $(\sqrt{2\pi}N\sigma)^{-2}$. For numerical reasons, we use the sum over the absolute values of the kernel matrix :

$$\mathcal{K}_{GCN}(x, y) = \frac{\mathcal{K}_{GC}(x, y)}{\sum_k \sum_l |\mathcal{K}_{GC}(k, l)|}, \quad k, l \in [-B : B]$$

4.4 Post Processing and Peak Picking

In this section, we explain how we can further improve the novelty curve and how a novelty curve is transformed into a set of boundaries through the peak picking procedure

4.4.1 Gaussian Cleaning

To get sharper peaks, we subtract a smoothed version of the novelty curve. The smoothed curve is computed by convolving the novelty curve $\mathcal{N}_{GCN}$ with a Gauss window function $w(n) = \exp\left(-\frac{1}{2}\left(\frac{n^2}{(\sigma B))^2}\right)\right), n \in [-B : B]$ with the window size $N_w = (2B + 1)$ and the standard deviation σ. We set B and σ to the same values as in the kernel matrix which was used to construct $\mathcal{N}_{GCN}$. Furthermore, we use the smoothed curve as a first threshold as we will only keep the non-negative values. The final novelty curve is then computed with

$$\mathcal{N}(i) = \max\left\{0, \mathcal{N}_{GCN}(i) - \left(\mathcal{N}_{GCN} * w\right)(i)\right\} \tag{4.9}$$

where $*$ is the convolution operator. In other words, we apply a high pass filter. The Gaussian window acts as low pass filter and subtracting a low pass filter from the original curve constructs a high pass filter [27]. In the remainder of the thesis, we use only Gaussian Checkerboard kernels and will denote $\mathcal{N}$ as the novelty curve from Equation (4.9).

4.4.2 Post Processing

To extract boundaries from the novelty curve, we need to find peaks in $\mathcal{N}$. A peak picker is a function $\Lambda : \mathbb{R}^N \to \mathbb{B}^N$ which transforms a sequence of real numbers into a binary sequence $PV \in \mathbb{B}^N$, where

$$\Lambda(N(i)) = \begin{cases} 1 & \text{if } \mathcal{N}(i) \text{ is peak} \\ 0 & \text{else} \end{cases}$$

We now only need to define what a peak is. We consider a point in the novelty curve to be a peak, if it the maximum value within a local neighborhood. Additionally we constrain the derivative of the novelty curve to be greater than zero at a potential peak.

4.4.3 Kernel Length, Smoothing and Normalization Impact

In this section, we propose a strategy to choose a good kernel length and show the response of the novelty curves when the kernel is chosen too large. We additionally illustrate the importance of using a normalized kernel. Finally, we show that the Gaussian kernel smooths the novelty curves. In Figure 4.5 a synthetic SDM with ideal checkerboard pattern is shown along with novelty curves computed with Gauss checkerboard kernels and box checkerboard kernels of different lengths. The box checkerboard and the Gauss checkerboard kernel are depicted twice, once with normalization and once without normalization. Each of the set of novelty curves is shown in two visualizations. The first visualization

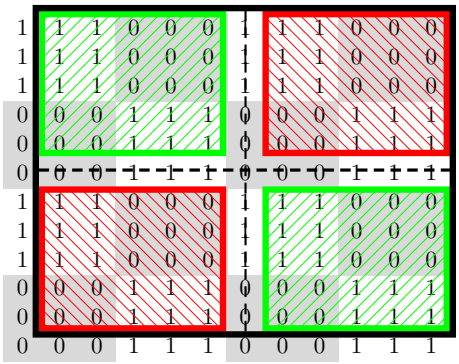


Figure 4.4. Illustration of a vanishing box checkerboard kernel on an excerpt of the matrix $\mathcal{D}$ in Figure 4.5. In the background, the values of $\mathcal{D}$ explicitly shown. The black rectangle in the foreground represents a BC kernel with $L = 11$. Within the green areas, the kernel has weights of one and within the red areas, the kernel has weights of minus one. The correlation of the kernel with the matrix at the indicated position is zero.

shows a standard line plot, where the discrete points of the novelty curves are linearly interpolated. In the line plot, some of the curves are not visible because they are overlapping with other curves. For this reason, we used a second visualization which shows the novelty curves in a matrix, where each row corresponds to a novelty curve. Another advantage of this visualization is, that it is much easier to distinguish the individual curves. One can see, that the height of unnormalized novelty curves $\mathcal{N}_{GC}$ and $\mathcal{N}_{BC}$ roughly correlates with the kernel length. The curve extracted with $L = 11$ in $\mathcal{N}_{GC}$ shows a higher value than the one with $L = 5$ and the curve computed with the smallest kernel length $L = 2$ nearly vanishes. This behavior is corrected in the normalized novelty curves $\mathcal{N}_{GCN}$. Note that the curves corresponding to the two smallest kernel lengths ($L = 3$ and $L = 5$) show exactly the same correlation values in $\mathcal{N}_{GCN}$ and $\mathcal{N}_{BCN}$. The block length of these kernels is smaller than 3, which fits perfectly within the segment size of $3s$, and thus results in a perfect correlation. The curve with $L = 11$ in $\mathcal{N}_{GCN}$ still shows a good correlation and $L = 35$ does still show a response, whereas in the box checkerboard examples $\mathcal{N}_{BCN}$ and $\mathcal{N}_{BC}$, the same curves vanish completely. The reason why $L = 11$ and $L = 35$ vanish in the box checkerboard examples is, that each of the blocks in the kernel matrix has the size corresponding to a region in $\mathcal{D}$ with the same number of white ($= 1$) and black blocks ($= 0$), which results in a zero correlation. The Gaussian checkerboard still shows some correlation for $L = 11$ and $L = 35$, because it does not weight all the points equally. The Gaussian checkerboard kernel basically adds a smoothing to the curve, due to its decaying shape.

We conclude that it is a good choice to set the block length of the kernel roughly to the average segment length. The kernel is then at maximum two times as long as an average segment. We have also seen that it is crucial to normalize the kernel matrix as the height of a novelty curve extracted with an unnormalized kernel matrix correlates to the kernel length.

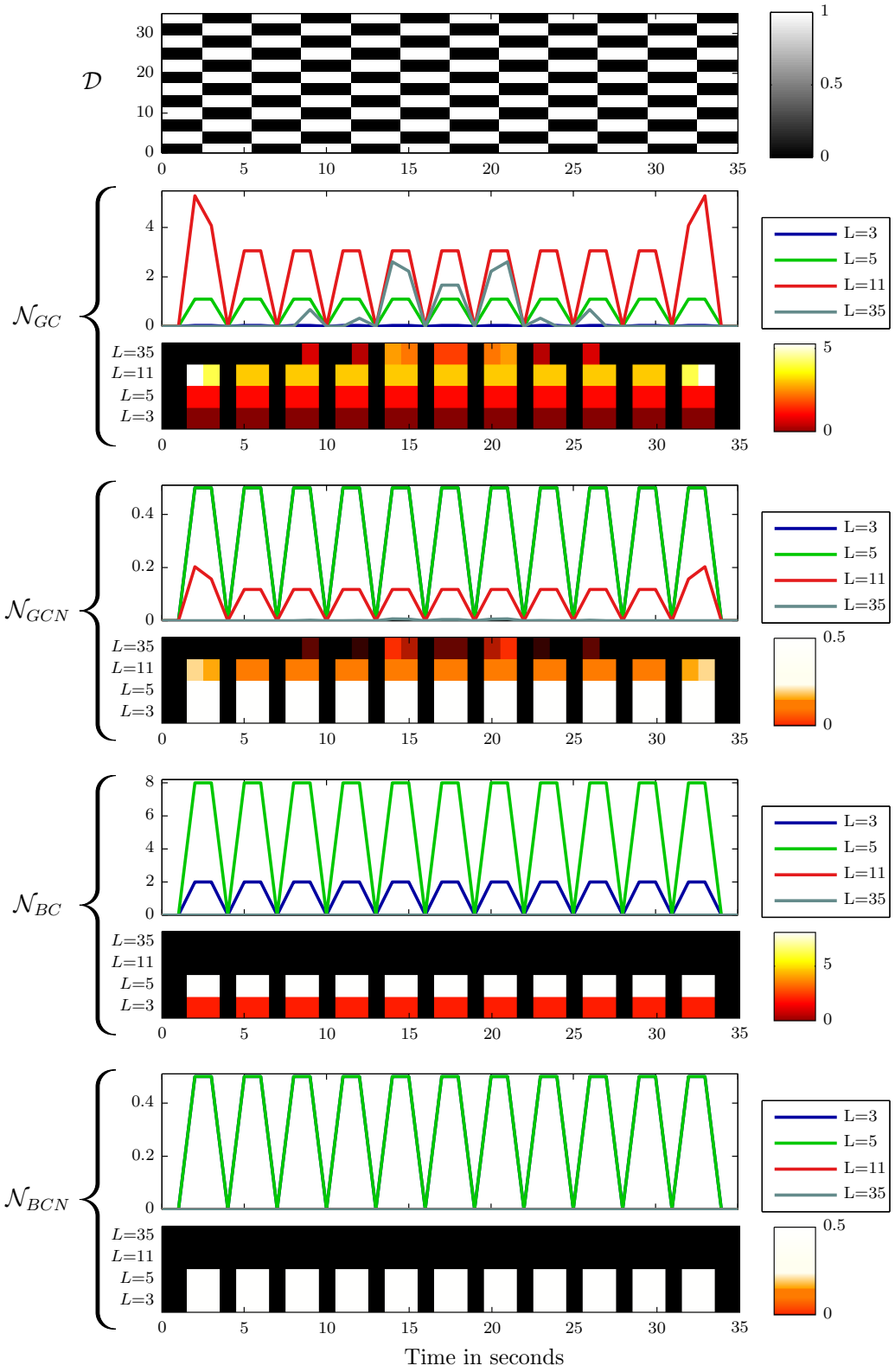


Figure 4.5. $\mathcal{D}$: Synthetic SDM. Each block in the checkerboard pattern corresponds to $3s$. Note that the matrix is still a square matrix and is only stretched to be aligned with the novelty curves. $\mathcal{N}_{GC}$: Novelty curve with Gauss checkerboard kernel. $\mathcal{N}_{GCN}$: Novelty curve with Gauss checkerboard kernel normalized with respect to L. $\mathcal{N}_{BC}$: Novelty curve with box checkerboard kernel. $\mathcal{N}_{BCN}$: Novelty curve with box checkerboard kernel normalized with respect to L.

4.5 Summary

The entire pipe of novelty extraction is summarized in Figure 4.6. First, the feature sequence is computed (a), then an SDM is computed on the feature sequence (b). In the next step, the novelty curve $\mathcal{N}_{GCN}$ (d) and its local average is computed (e). Finally, the local average is subtracted from the original novelty curve to get sharper peaks and the peak picker is applied (f).

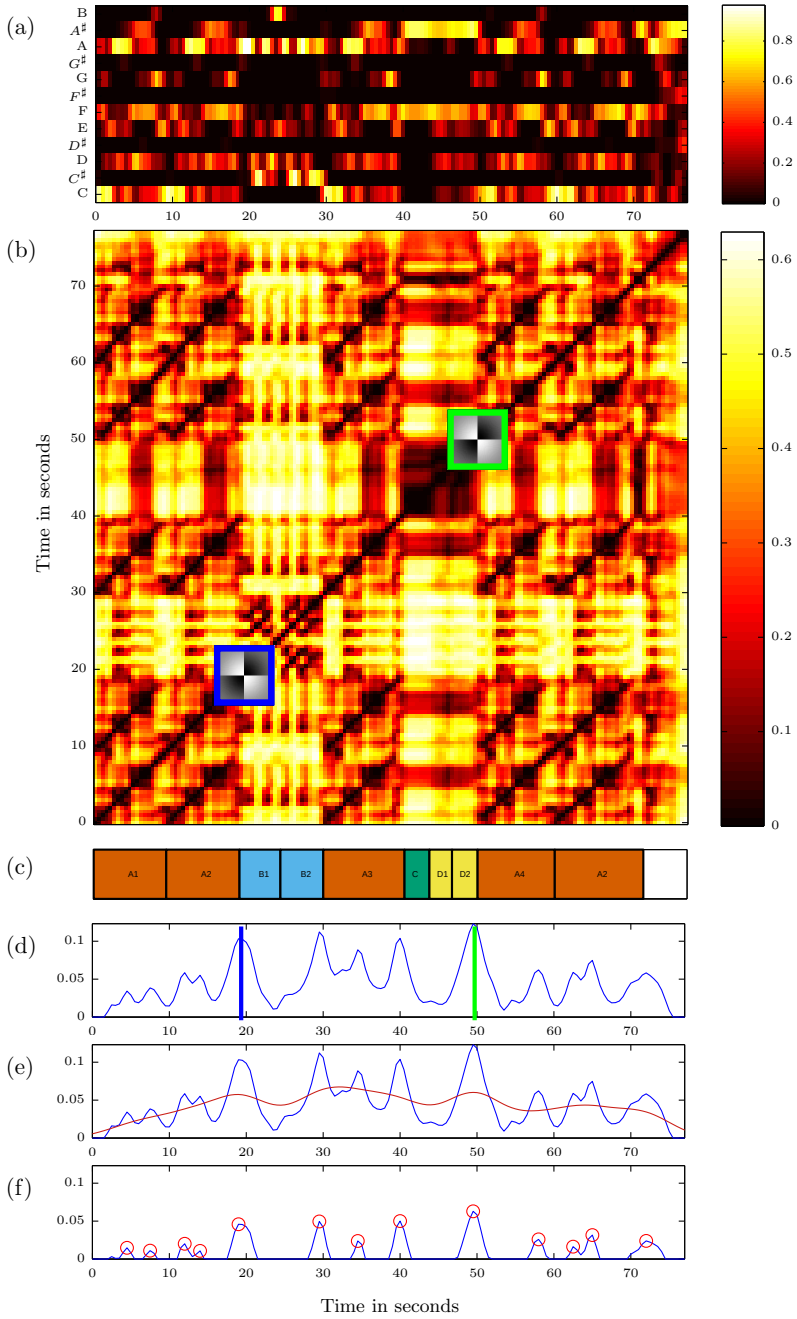


Figure 4.6. The figure shows the entire pipeline of the novelty detection algorithm on a recording of Chopin's Mazurka Op. 68 No. 3 performed by Ashkenazy in 1981. **(a):** CENS feature sequence. **(b):** SDM. The blue and green squares illustrate how the Gaussian checkerboard kernel is correlated along the main diagonal. **(c):** Structural annotation. The white region at the end corresponds to silence at the of the recording. **(d):** Raw novelty curve. The blue and green line correspond to the position of the Gaussian checkerboard kernels in the SDM. **(e):** Raw novelty curve (blue). The curve in dark red shows the local average with respect to the kernel length L. **(f):** Final novelty curve (blue) where the local average was subtracted from the original novelty curve. The red circles show the result of the peak picking.

Tests test tests.

Heinz von Foerster

5 Evaluation and Datasets

In this chapter we will introduce the evaluation metrics for later usage in our experiments. It is important to have an objective measure to evaluate the performance of an algorithm. Furthermore, we will give a short overview of the dataset which we will use for the experiments.

5.1 Boundary Evaluation

The boundary detection task can be modeled as a two class classification problem with the classes labelled as *boundary* and *non-boundary*. In the following, we will define *precision* P, *recall* R and *F-measure* F, which are standard measures from information retrieval. Let $\mathcal{S}_{GT}$ denote the set of boundaries from a reference segmentation, which must not necessarily be a human produced ground truth, and $\mathcal{S}_E$ denote an estimate of the boundaries.

$$P = \frac{|\mathcal{S}_{GT} \cap \mathcal{S}_E|}{|\mathcal{S}_E|} \qquad R = \frac{|\mathcal{S}_{GT} \cap \mathcal{S}_E|}{|\mathcal{S}_{GT}|} \qquad F = 2\frac{PR}{P+R} \tag{5.1}$$

Precision measures the fraction of the number of correct results over the number of computed results. If all computed results are correct, $P = 1$. Recall measures the fraction of correct results over the number ground truth annotations. A high recall value does not necessarily imply a good performance. If our algorithm predicted all points in time to be boundaries, the intersection $\mathcal{S}_{GT} \cap \mathcal{S}_E = \mathcal{S}_{GT}$ between the set of predicted boundaries $\mathcal{S}_E$ and the reference boundaries $\mathcal{S}_{GT}$ would be equal to $\mathcal{S}_{GT}$ which results in $R = 1$, whereas $P \approx 0$. As it does not make always sense to look at isolated precision or recall values, one computes the harmonic mean between them, which is the F-measure. P, $R \in [0, 1]$ from which follows that $F \in [0, 1]$

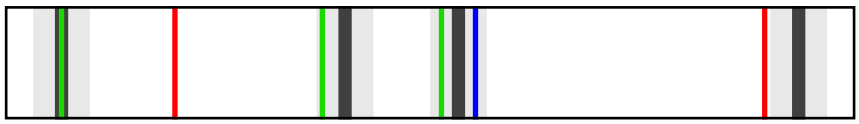


Figure 5.1. The thick dark gray lines indicate a ground truth (GT) boundary. The light gray background indicates the tolerance region around a GT boundary. A green line indicates a correct prediction and a red line indicates a wrong prediction. A blue line indicates a prediction which lies within a tolerance region together with another prediction which already has been counted correct. Only one prediction within a tolerance region can be considered to be correct.

The estimated boundaries do not always match exactly to the boundaries in the reference. The reason for this can be a coarse feature resolution or the algorithm is simply not accurate enough. In structural audio segmentation, where we segment an audio file into rather big chunks, small deviations from the reference segmentation should still be considered as a valid output. As stated in [26], common tolerance ranges are $\pm 0.5s$ and $\pm 3s$. Later on, we will work on a musical time axis, i.e. measures and beats and use a tolerance of ± 1 measure. Figure 5.1 gives an illustration how we compare a ground truth (GT) boundary annotations with computed results. Note that there are two boundaries in the tolerance region of the third GT boundary. The blue line indicates, that the second prediction is wrong because another boundary within the same tolerance region has already been counted as correct. If we would count both boundary values as correct, we would distort the P, R, F to attain higher values which could even be greater than one. We will at most count one computed boundary within one GT tolerance region to be correct.

Furthermore, we will exclude the boundaries at the very starting and very ending of a piece because of possible synchronization errors at the start and end of a piece. This is due to silence in the start and end segments of a music recording, which results in low cost regions of the cost matrix C where the DTW algorithm tends to choose a random path, see Chapter 3 for details.

5.2 Evaluation on the Musical Time Axis

The final evaluation will be done on the musical time axis. To this end, we introduce a method of assigning real numbers to a position in a measure. To annotate a piece of music on the measure level, we developed the MetaMIDI annotation format. In this section, we briefly review some basic musical vocabulary and definitions and then introduce the annotation format.

5.2.1 Basic Musical Vocabulary

A *note value* defines the relative duration of a note. Examples of note values are quarter notes (♩), eighth note (♪) and so on. The *time signature* specifies the relative duration of a measure with respect to a reference note value. It consists of two numbers which are usually printed as fraction at the beginning of a musical score. The denominator specifies

the reference note value which is often also the duration of a beat, e.g. 4 for a quarter note. The nominator specifies how many of these note values make a full measure. An *upbeat measure* (also called pickup measure or anacrusis), is a measure at the beginning of a piece which does not contain the full number of note values given in the time signature. Often its note value is the half of the duration of a beat.

In musicology, the first measure is assigned the number one. The measure counting starts at the first full measure. A measure is full when its duration is equal to the number of note values specified in the time signature. In the following we explain how we assign determine positions within a musical measure.

5.3 Musical Measure Counting

We distinguish between a *measure number* and a *measure position*. A *measure number* is an integer referring to a measure as a whole whereas a *measure position* is a real number which refers to a position within a measure. Let $m \in \mathbb{N}$ be a measure number. Then the real number $m.000$ refers to the start position of measure m. The other measure positions are dependent on the time signature. The position in the first measure before the second quarter note in a piece with 3/4 time signature is 1.333. In pieces of music with an upbeat measure, the measure position before the first note in the upbeat measure refers to the start of the piece. The measure number of an upbeat measure is zero, but all its measure positions are greater than zero. The measure position which defines the start of the upbeat measure is determined by the time signature of the piece and the theoretical beat position of the upbeat note in a full measure. We analogously use a *beat number* to refer to a beat as a whole and a *beat position* to refer to the exact position of a beat. Given a piece of music with the time signature n/d and $n, d \in \mathbb{N}_{>0}$, a measure position is computed as following:

$$QuartersPerMeasure = 4 \cdot \frac{n}{d} \tag{5.2}$$

$$MeasurePosition = m + \frac{q}{QuartersPerMeasure} \tag{5.3}$$

where m is the current measure number and $q \in \mathbb{R}$ refers to the current beat position in quarters.

The example of a musical score in Figure 5.2 has an upbeat measure which contains one eighth note. The time signature is 3/4 with $n = 3$ and $d = 4$. To compute the measure position before the eighth note in the upbeat measure, we first determine its beat position in quarter notes in a theoretical full measure. The last eighth note in a measure with 3/4 time signature is on beat position 2.5. Thus, the measure position x_0 at the beginning of the piece is $x_0 = 0 + \frac{2.5}{3} \approx 0.833$

5.3.1 MetaMIDI

We developed the MetaMIDI annotation format which stores meta information such as the number of measures, the length of a musical beat, the time signatures of a piece of

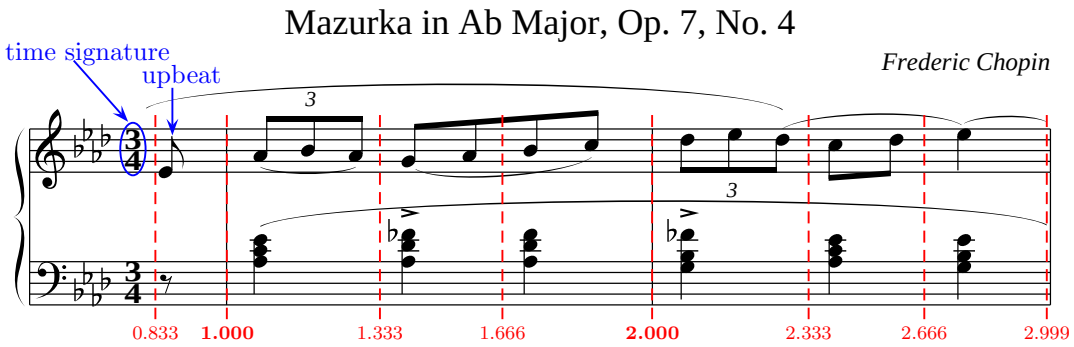


Figure 5.2. Excerpt of Chopin's Mazurka Op. 7 No. 4 with measure positions and upbeat measure. The red dashed lines with the number in red indicate the measure positions.

music and structural annotations. One of the main goals of developing MetaMIDI was to provide the possibility of assigning the musically correct measure numbers even in the presence of an upbeat at the beginning of a piece of music. To this end, we store the length of the upbeat measure in quarter notes in the annotation file. With this information, we can then compute the correct measure positions at a given beat position. The reason why this is not possible by just using MIDI files is that there is no way to specify an upbeat in a MIDI file. Sometimes people use the time signature information to encode an upbeat MIDI files. This only helps to determine the correct start positions of the subsequent measures, but it still results in a wrong measure numbering. The MetaMIDI format and its fields are described in Appendix A.

5.4 Dataset

We conduct our experiments on the Mazurka dataset [8, 25]. The Mazurka Project collected over 2700 recording for 49 Chopin Mazurkas. The Chopin Mazurkas are short piano compositions wit a 3/4 time signature. For every Mazurka in the dataset, there is reference MIDI version, a MetaMIDI file with structural annotations and on average 57 audio recordings. Table 5.1 shows an overview of the Mazurka dataset

Piece	Versions	$\#M$	$\varnothing\ dur(s)$	$\#B$	$dur(s)/\#M$	$\#M/\#B$
06-1	49	112	173.60	7	1.55	16.00
06-2	51	96	152.12	9	1.58	10.67
06-3	47	98	113.12	12	1.15	8.17
06-4	46	40	52.28	9	1.31	4.44
07-1	55	104	137.05	9	1.32	11.56
07-2	51	118	197.11	14	1.67	8.43
07-3	65	105	146.80	11	1.40	9.55
07-4	43	60	72.10	7	1.20	8.57
07-5	46	20	43.68	12	2.18	1.67
17-1	52	100	138.94	10	1.39	10.00
17-2	55	68	125.41	3	1.84	22.67
17-3	51	168	265.58	10	1.58	16.80
17-4	93	132	256.82	10	1.95	13.20
24-1	61	96	170.98	10	1.78	9.60
24-2	66	120	137.86	16	1.15	7.50
24-3	55	79	125.57	6	1.59	13.17
24-4	76	186	282.60	20	1.52	9.30
30-1	50	53	99.99	4	1.89	13.25
30-2	60	64	84.03	7	1.31	9.14
30-3	62	111	169.76	10	1.53	11.10
30-4	65	139	232.11	14	1.67	9.93
33-1	55	48	101.96	4	2.12	12.00
33-2	66	143	142.49	16	1.00	8.94
33-3	55	48	112.40	3	2.34	16.00
33-4	74	224	299.81	19	1.34	11.79
41-1	56	139	206.39	14	1.48	9.93
41-2	63	68	144.64	7	2.13	9.71
41-3	40	78	76.51	13	0.98	6.00
41-4	45	74	122.29	8	1.65	9.25
50-1	50	104	144.09	6	1.39	17.33
50-2	58	127	184.12	10	1.45	12.70
50-3	74	208	306.26	10	1.47	20.80
56-1	42	204	262.27	13	1.29	15.69
56-2	54	92	108.11	8	1.18	11.50
56-3	56	220	354.72	15	1.61	14.67
59-1	63	142	235.77	9	1.66	15.78
59-2	63	111	158.58	4	1.43	27.75
59-3	66	154	207.91	11	1.35	14.00
63-1	46	102	131.63	9	1.29	11.33
63-2	65	56	113.21	4	2.02	14.00
63-3	88	76	128.98	8	1.70	9.50
67-1	43	60	75.58	7	1.26	8.57
67-2	41	72	118.90	6	1.65	12.00
67-3	47	56	91.44	3	1.63	18.67
67-4	60	112	170.52	6	1.52	18.67
68-1	46	84	100.17	11	1.19	7.64
68-2	64	84	173.50	11	2.07	7.64
68-3	51	60	101.17	9	1.69	6.67
68-4	63	63	148.04	7	2.35	9.00
Σ	2793	5078	7698.95	461	76.80	582.22
$\varnothing$	57.00	103.63	157.12	9.41	1.57	11.88

Table 5.1. Overview of the Mazurka dataset. $\#M$ denotes the number of measures, $\#B$ denotes the number of boundaries.

Act always so as to increase the number of choices.

Heinz von Foerster

6 Cross-Version Approach

In this chapter, we bring together the techniques introduced in the previous chapters. We will build a cross-version pipeline to combine the results of the novelty detection from different versions of a piece of music into one common result. Recall our main assumption, that the combination of a set of versions of a piece of music reveals more information about the piece of music itself than the individual version. One single performance is biased towards the characteristics of a specific performer. Another goal of the cross-version approach is to establish a new evaluation framework to compare the outcome of boundary detection algorithms on many versions of a piece. As we will deal with classical music, we will also transform the data from the physical time axis to the musical time axis. This makes the inspection of the results easier, as we will be able to look at a curve and find the corresponding musical measure position in the sheet music.

The chapter is structured as follows: In Section 6.1, we explain how a novelty curve can be transformed onto the musical time axis. Section 6.2 introduces the basics of the cross-version analysis. In Section 6.3 we discuss different error sources and in Section 6.4 we discuss how the agreement within a set of novelty curves can be measured.

6.1 Warping Novelty Curves

In this section, we explain how the novelty curves of different versions of the same piece of music are transformed onto the musical time axis. This procedure has two advantages. First, it makes the comparison of results from different versions easier. Second, when the results are on the musical time axis, it is easier to compare them with the musical score. Let K be the number of versions of a piece of music in the database. Then $k \in [1 : K]$ denotes the k^{th} version.

We denote $\mathcal{N}^{ref}$ as the novelty curve of a reference version and $\mathcal{N}^{k}$ as the novelty curve

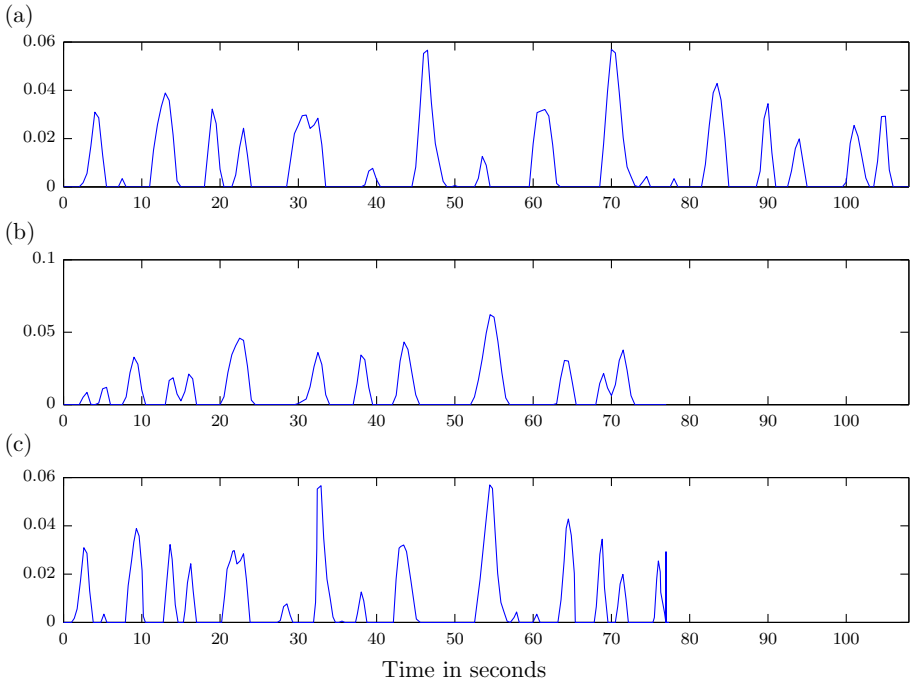


Figure 6.1. Novelty curves on Mazurka Op. 68 No. 3 with $L = 33$. **(a)**: Novelty curve $\mathcal{N}^k$, performed by Bacha. **(b)**: Novelty curve $\mathcal{N}^{ref}$. **(c)**: $\hat{\mathcal{N}}^k$ (Novelty curve $\mathcal{N}^k$ warped to the reference version).

corresponding to a different version of the same piece of music as the reference version. The novelty points $\mathcal{N}^{ref}(i)$ are defined over the index set $i \in [1 : N]$ and $\mathcal{N}^k(j)$ is defined over the index set $j \in [1 : M]$. As reference, we use a MIDI version. The advantage of using a MIDI version is, that we can use the note onset informations to locate the musical beat positions.

To compute the physical time corresponding to the novelty point $\mathcal{N}^v(j)$, we use the function

$$t = \frac{(j-1)}{sr} \tag{6.1}$$

where sr is the sample rate of the novelty curve. Note that $sr = fr = 2$ Hz where fr is the feature rate of the features which were used in the computation of the SDM. To transform the points in time $t(j)$ of $\mathcal{N}^k(j)$ to the reference version we apply the function

$$\hat{t}(j) = timewarp(t(j), p, sr_p) \tag{6.2}$$

where $j \in [1 : M]$, p denotes the warping path and sr_p is the sampling rate of the warping path. We define the warped novelty curve

$$\hat{\mathcal{N}}^k(\hat{t}\,(j)) := \mathcal{N}^k(j) \tag{6.3}$$

on the set of points in time $\hat{t}([1 : M])$. Note that $\hat{t}(j) < \hat{t}(j+1)\,\forall j \in [1 : M-1]$, which follows from the strict monotonicity of the *timewarp* function. Figure 6.1a shows an example of a novelty curve $\mathcal{N}^k$ on its original time axis. Figure 6.1b shows the reference

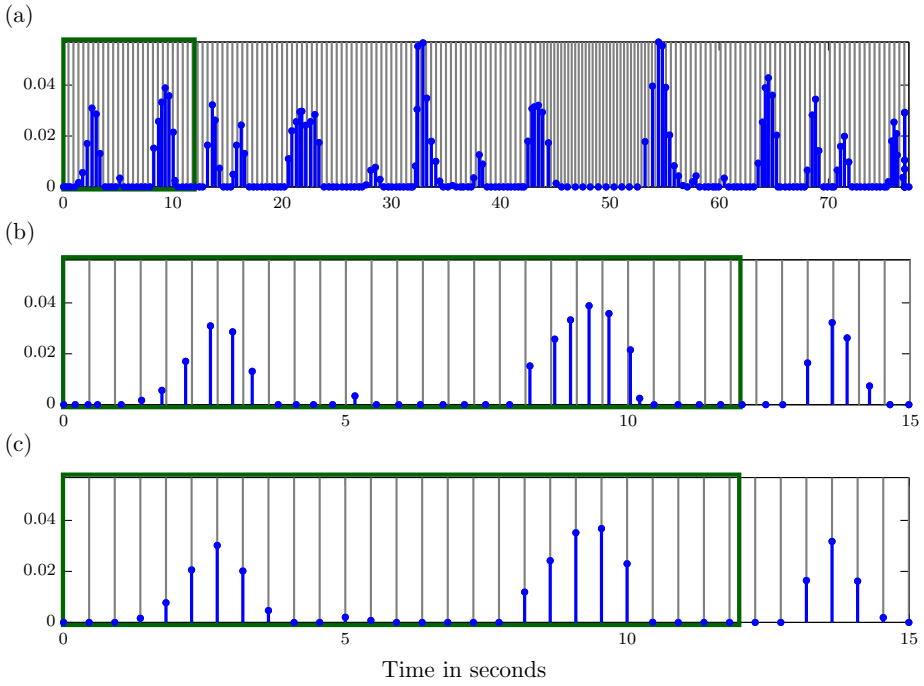


Figure 6.2. Warped novelty curves. The gray lines mark the beat positions in the MIDI. The blue lines mark the values of the novelty curves. **(a)**: Novelty curve $\hat{\mathcal{N}}^k$ **(b)**: Zoom in of $\hat{\mathcal{N}}^k$ **(c)**: $\mathring{\mathcal{N}}^k$ on beat positions.

curve $\mathcal{N}^{ref}$ which was computed on a MIDI version. Figure 6.1c shows the warped novelty curve $\hat{\mathcal{N}}^k$. One can clearly see from Figure 6.1b and c, that the two curves have a very similar shape.

In a further step, we resample the novelty curves to the musical time axis by extracting the beat positions from the MIDI version. Our goal is to have one novelty value at each beat position. First we extract the beat positions from the MIDI. We denote the set of musical beats as $\mathcal{B} = [1 : W]$. The function $\pi : \mathcal{B} \to \mathbb{R}$ maps a beat to its physical time position in the MIDI version. As we are working in a discrete setting, the novelty curves are only defined on a discrete index set. Figure 6.2 illustrates this fact, by visualizing the values of the warped novelty curve $\hat{\mathcal{N}}^k$ as vertical lines at the points in time $\hat{t}(j)$ with $j \in [1 : M]$. The novelty curve in Figure 6.2 is the same as shown in Figure 6.1c. The gray lines in the plots mark the beat positions. Figure 6.2b shows a zoomed excerpt of the novelty curve. One can see, that the points of the curve are often located between the beats. To transform the points of the novelty curve $\hat{\mathcal{N}}^k$ to the positions $\pi(b)$ we apply a linear interpolation:

$$\mathring{\mathcal{N}}(b) := \hat{\mathcal{N}}\left(\hat{t}(j-1)\right) + \left(\pi(b) - \hat{t}(j-1)\right) \frac{\hat{\mathcal{N}}\left(\hat{t}(j)\right) - \hat{\mathcal{N}}\left(\hat{t}(j-1)\right)}{\hat{t}(j) - \hat{t}(j-1)} \tag{6.4}$$

where

$$j = \begin{cases} 2 & \text{if } \hat{t}(1) > \pi(b) \\ \underset{\ell \in I}{\operatorname{argmin}} \ \pi(b) - \hat{t}(\ell - 1) & \text{if } I \neq \emptyset \\ M & \text{if } \pi(b) > \hat{t}(\ell)\,,\ \forall \ell \in [1:M] \end{cases} \tag{6.5}$$

and $I := \{\ell \,|\, \hat{t}(\ell - 1) \leq \pi(b) < \hat{t}(\ell) \wedge \ell \in [2 : M]\}$. Note that Equation (6.5) and Equation (6.4) are formulated, such that a linear extrapolation is applied in the case a linear interpolation is not possible. The linear extrapolation is formulated in the first and last case of Equation (6.5).

Figure 6.2c shows the resulting novelty curve on the beat positions.

6.2 Cross-Version Analysis

In the previous section we explained how to warp a novelty curve to the beat positions of a reference MIDI. In this section we apply this method to a set of versions of the same piece of music and show how the results can be combined. Let $\mathcal{M}$ be a piece of music with K versions. We define the *novelty matrix*

$$\mathcal{V}(k, b) = \mathring{\mathcal{N}}^k(b) \tag{6.6}$$

as the matrix containing all novelty curves for the versions $k \in [1 : K]$ on the beat level with $b \in \mathcal{B}$. Having the novelty curve on the beat level we can easily compute their measure position. Figure 6.3d visualizes an example of the matrix $\mathcal{V}$ where the time axis is indicated in measures.

We use a matrix visualization as it has the benefit that one can better distinguish the individual novelty curves. To show the advantage of analyzing the novelty curves on the same time axis, we plotted the same curves on their original time axis in one plot in Figure 6.3a. In this visualization, it is nearly impossible to compare the different curves and to see a common structure. Figure 6.3b shows the data of the matrix $\mathcal{V}$ visualized in one plot. Here, we already see that the majority of the curves have the same shape but again, it is hard to distinguish the individual curves.

To combine the information of all novelty curves into a *fusion curve*, we compute an *average novelty curve* by applying the arithmetic mean over the columns of the novelty matrix:

$$\mathring{\mathcal{N}}_{avg}(b) = \frac{1}{K} \sum_{k=1}^{K} \mathring{\mathcal{N}}^k(b) \tag{6.7}$$

An example of an average novelty curve is shown in Figure 6.3. The higher the value at a beat position b in the average novelty curve, the more versions agree at this position. The deviation of individual curves is suppressed in the average curve which makes the results more robust. By suppressing the "noise" from individual performances we can draw a clearer picture of a piece of music. In the following section we will discuss several error sources in the cross version pipeline and explain how we can measure the mutual agreement within a set of recordings.

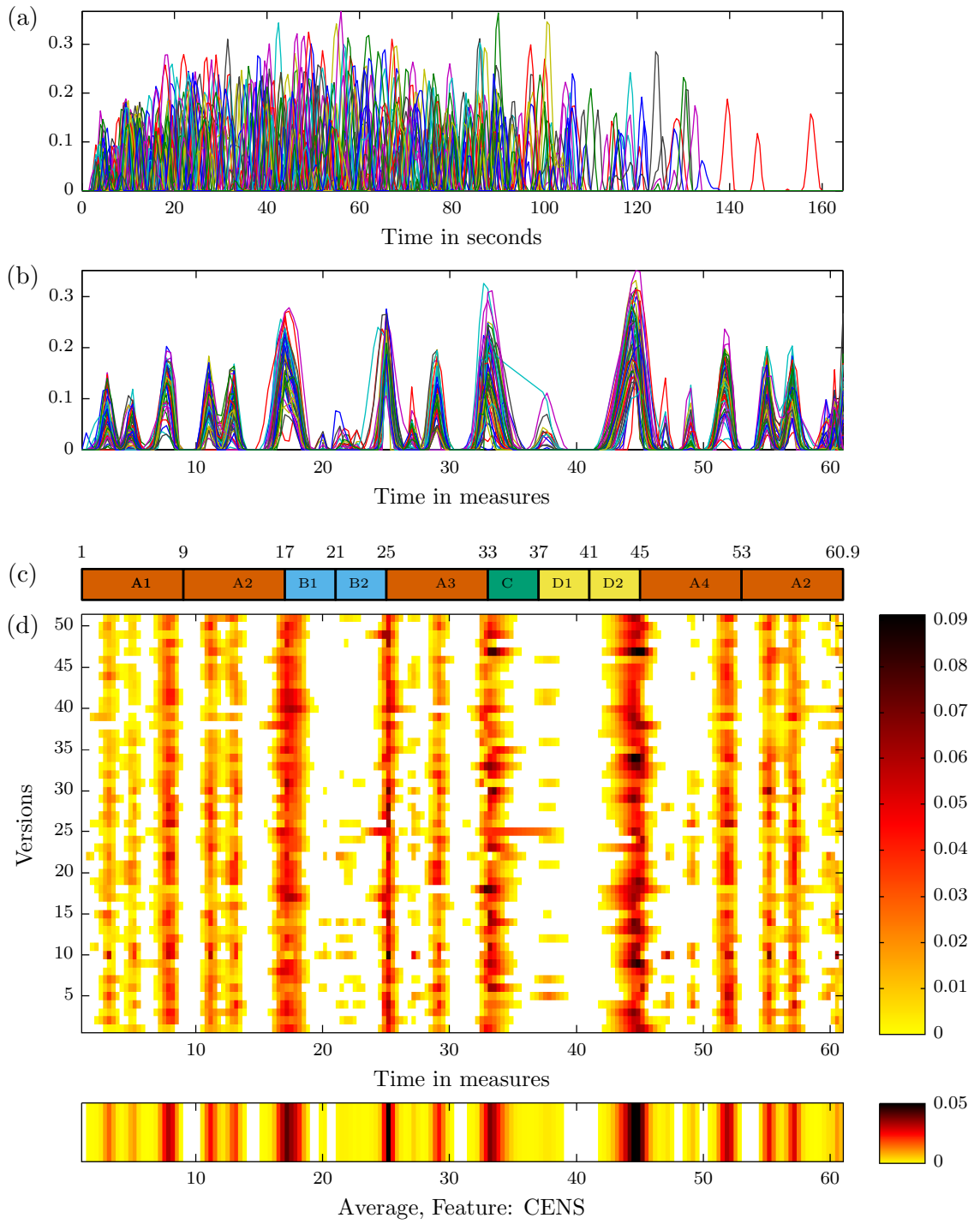


Figure 6.3. Example of all versions of Chopin's Mazurka Op. 68 No. 3. All novelty curves were computed on CENS features with a feature rate of 2 Hz and a kernel length L=33 **(a)**: Novelty curves on their original time axis, **(b)**: Novelty curves on the musical time axis. One can see, that most of the curves have the same shape. **(c)**: Ground truth annotation, **(d)**: Novelty matrix on the same data as (b), but in a matrix visualization..

	Index	1	2	3	4	5
Structure	Version 1	A	B	C		
	Version 2	A	B	A	B	C

	Index	1	2	3	4	5
Warping Path	Version 1	1	2	2	2	3
	Version 2	1	2	3	4	5

Table 6.1. Illustration how structural outliers cause synchronization errors.

6.3 Possible Error Sources

6.3.1 Synchronization Errors and Structural Outliers

There are two types of synchronization errors. First, there are the synchronization errors at the beginning and the end of a recording. And second, there are *structural synchronization errors* which are caused by structural outliers. Those errors are in fact not a malfunctioning of the synchronization procedure, but a problem in the dataset. There are two types of structural outliers that can be identified:

1. Performances which omit parts. Most commonly, repetitions are omitted. But there exist also some performers who deliberately interpret a piece and omit sections or repeat sections where no repetition is notated in the score. These structural deviations are visible in the novelty if there is a peak in the region of the structural deviation. Those peaks are visible as blurred regions in the novelty matrix $\mathcal{V}$, see Figure 6.4. This pattern can be explained as follows: When two versions of a piece of music are globally synchronized but do not share the same structure at some points, some points in time from one of the versions are assigned to many points in time of the other version. Lets consider the example of one version which follows the structure 'A B C' and one which has a repetition of the first two parts 'A B A B C'. The synchronization algorithm would probably assign the repetition of the second version to the B part of the first one. The example is illustrated in Table 6.1 together with a hypothetical warping path.

2. Performances which belong to another piece of music and are wrongly assigned to set of versions of a different piece. This is one of the problems which occur in the dataset. If the number of wrongly assigned versions is not too high, then this does not impose a big problem.

An example of a novelty matrix on Op. 68 No. 4 which shows the two types of structural deviations described above is shown in Figure 6.4. This example is an extreme case, since the majority of the versions suffer from synchronization errors as the structure of the underlying piece is not well defined. At the end of the score it has the performance instruction "D. C. al segno senza fine". Senza fine means "without" end and there are

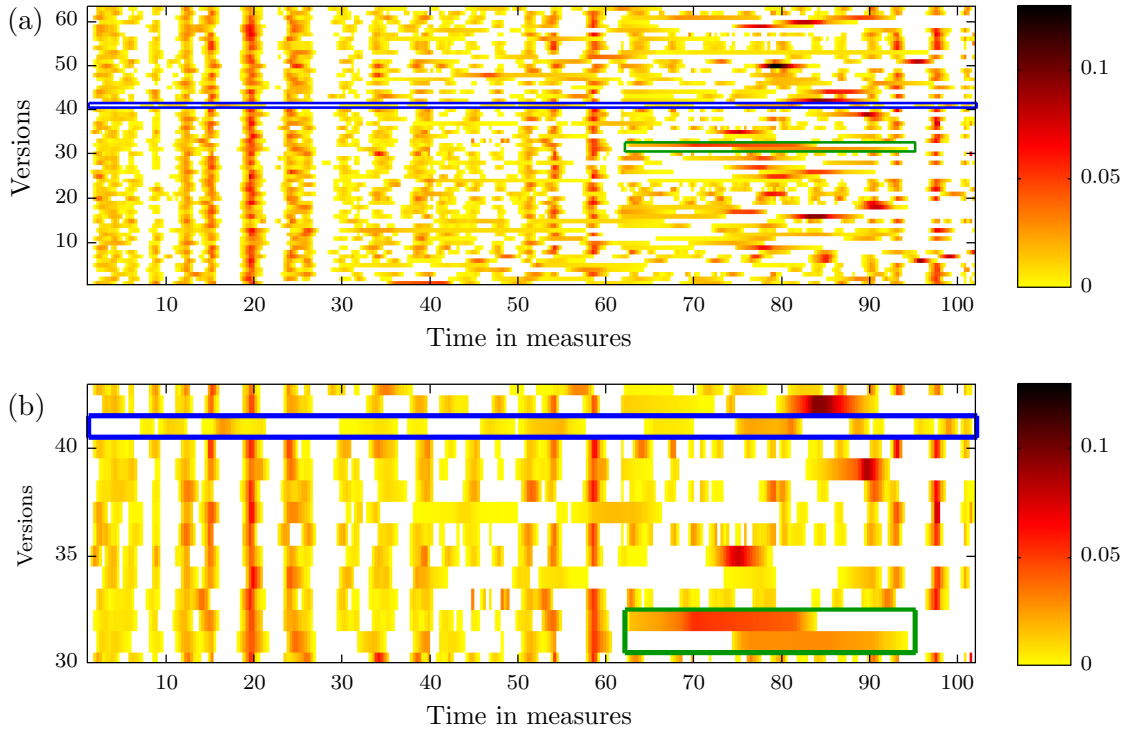


Figure 6.4. (a): Novelty matrix of Mazurka Op. 68 No. 4 computed on CENS features. **(b)**: Excerpt from matrix (a). The green box marks two versions which show a blurred peak that is caused by structural synchronization error. The blue box marks a novelty curve of a version which has been wrongly assigned to the set of recordings for Op. 68 No. 4 and actually belongs Op. 67 No. 1.

some versions of the musical score with an end mark (fine) and some without[1]. Thus, a pianist has a lot of freedom in interpreting the piece and to decide where to end it.

6.3.2 Feature Resolution

Figure 6.5 shows the novelty curves of the longest and the shortest version from the set of recordings of Op. 68 No. 3 on the beat level. The curve in Figure 6.5a and b were computed on a recording played Rubinstein with a total duration of 72s. The novelty curve in Figure 6.5c was computed on a recording played by Cortot with a total duration of 165s. Considering only the total durations of the recordings, they differ by a factor of 2.3. Approximately we can say, that the recording played by Rubinstein was played with twice as many beats per minute as the recording played by Cortot. Remember that the audio features were extracted with a feature rate of 2 Hz. Thus, one feature in the shorter recording captures roughly twice as many musical events as a feature in the longer recording. This is also reflected in the novelty curves when the same kernel length is used. The curves in Figure 6.5a and c were computed with $L = 49$. We can clearly see, that the curve in Figure 6.5c shows much more novelty peaks as the curve in Figure

[1] Different versions of the score can be found at http://imslp.org/wiki/Mazurkas,_Op.68_(Chopin,_Frédéric)

6.5a. As the novelty curves are computed on a SDM which is derived from the extracted features, the SDM corresponding to a shorter recording is smaller than a SDM of a longer recording. With the feature resolution of 2 Hz, we get 144 features from the Rubinstein recording and 330 features from the Cortot recording. To better illustrate this effect, the SDM corresponding to the Rubinstein recording is depicted in Figure 6.5d and the SDM computed on the Cortot recording is depicted in Figure 6.5e. As expected, the SDM in Figure 6.5e is much finer grained than the one in Figure 6.5d.

This problem could be solved by using an adaptive feature extraction, where the window size is adapted to the local tempo of the recording. Another approach would be to choose the kernel length for the novelty computation adaptively depending on the local tempo. A very simple, but not as accurate solution would be to choose the feature rate depending on the total duration of a piece.

6.4 Cross Interpretation Evaluation

6.4.1 Measuring the Agreement across Versions

In Figure 6.3d, we can see that there is some variation across the novelty curves of the different versions. Most curves follow a common shape, but some significantly deviate from the others. There is also some variation in the general amplitudes of the curves. To measure the general agreement within a set of curves, we could simply compute their deviation from the average curve. This would be a suitable approach, if the peaks in the curves would have nearly the same amplitude. To make the measure invariant to high variations in amplitude, we first compute a description of the rough shape of a curve by applying the sign function on the discrete derivative of the novelty curve:

$$rs(\mathring{\mathcal{N}}^k(b)) := \operatorname{sgn}\left(\mathring{\mathcal{N}}^k(b+1) - \mathring{\mathcal{N}}^k(b)\right) \tag{6.8}$$

The function rs describes the progression of a novelty curve over time with only three values. The rough shapes of all novelty curves are store in the rough shape matrix

$$RS(k,b) = rs(\mathring{\mathcal{N}}^k(b)) \tag{6.9}$$

To measure the agreement in a novelty matrix, we first apply the function rs on each individual novelty curve and on the average novelty curve. Then we compute the *absolute deviation* of every single novelty curve from the average novelty curve. To finally compute the agreement within a whole dataset we compute the *mean absolute deviation*.

$$MAD(\mathcal{V}) = \frac{1}{K \cdot (W-1)} \sum_{k=1}^{K} \sum_{b=1}^{W-1} \left| rs\left(\mathring{\mathcal{N}}^k(b)\right) - rs\left(\mathring{\mathcal{N}}_{avg}(b)\right) \right| \tag{6.10}$$

where $\mathcal{V}$ is a novelty matrix, K is the number of versions, W is the number of beats. The MAD measure is normalized such that it lies within $[0,1]$. The lower the MAD value, the more versions in a set do agree. A high MAD value is an indication for a lot of disagreement in a novelty matrix.

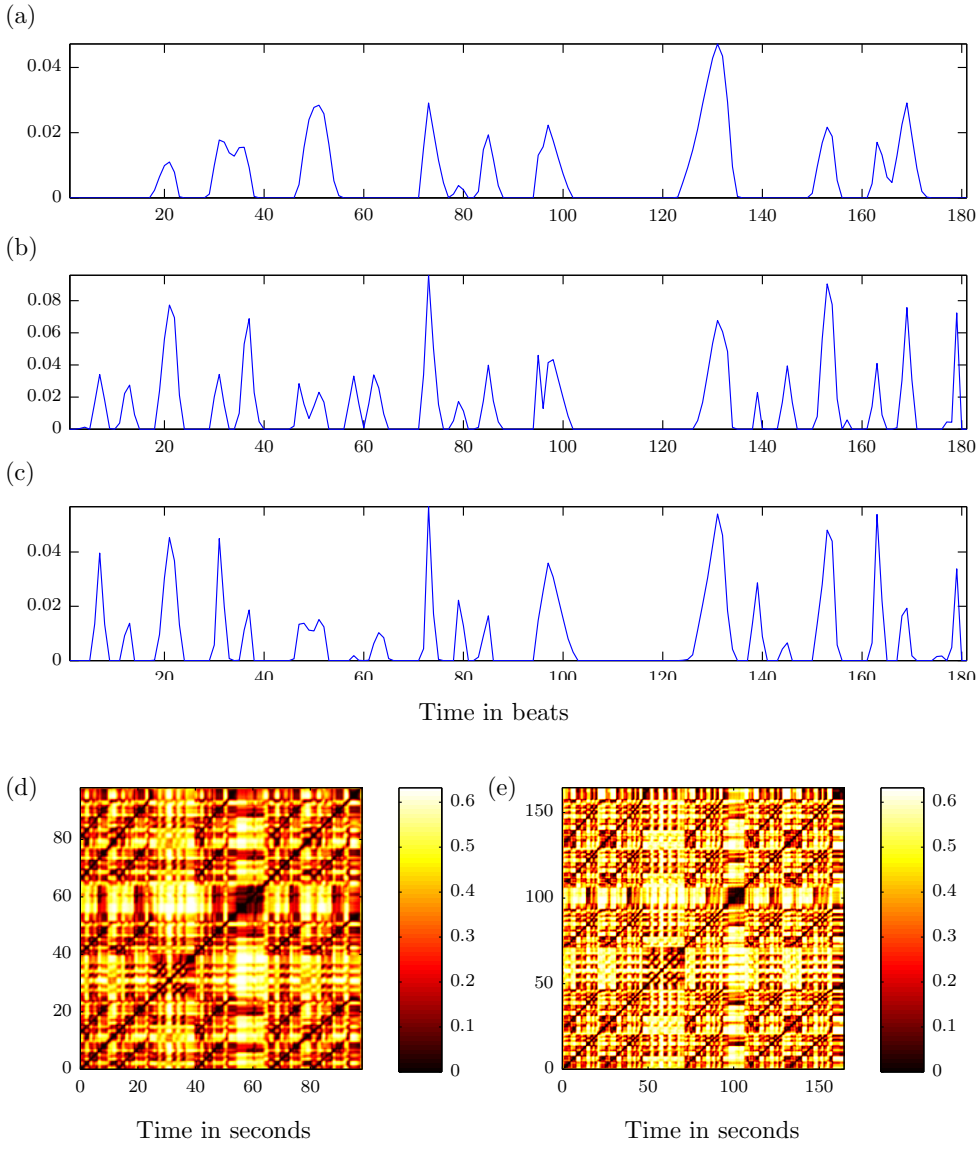


Figure 6.5. Novelty curves on Chopin's Mazurka Op. 68 No. 3. (a) and (b) are computed on the same version but with different kernel lengths. **(a)**: $L = 49$. Novelty curve of a version performed by Rubinstein which has a total duration of $72s$ **(b)**: $L = 17$ on same version as (a) **(c)**: $L = 49$. Novelty curve of a version performed by Cortot which has a total duration of $165s$ **(d)**: SDM corresponding to Rubinstein version, from which the curve in (a) and (b) were derived. **(e)**: SDM corresponding to Cortot version, from which the curve in (c) was derived.

7 Experiments

In this chapter, we present experimental results. First we present an experiment on different strategies to combine our two tempo features in Section 7.1. In Section 7.2, we compare the performance of the boundary detection on novelty curves from individual versions with the performance on the fusion novelty curve.

7.1 Combining Tempo Features

In Section 2.4 we introduced two methods to extract cyclic tempogram features, ctF extracted from a Fourier tempogram and ctACF extracted by the autocorrelation function. Both tempograms emphasize different aspects, i.e. the Fourier tempogram emphasizes harmonics and the autocorrelation tempogram emphasizes subharmonics [11]. Our goal is to fix a merged cyclic tempogram feature . First, we will investigate the impact of the dimensionality of the tempo feature on the performance values in the novelty detection algorithm. We tested 10, 15 and 30 dimensional tempo features on a 2 Hz resolution and three Gaussian checkerboard kernels with the lengths $L = 17, 33, 49$ and $\sigma = 0.5$.

Theoretically, the kernel covers $\lfloor L/2 \rfloor$ seconds before and and after the feature $\boldsymbol{v}_i, i \in [1 : N]$. But due to the decay of the Gaussian, this reduces roughly to the half.

The 10- and 15-dimensional tempo features were computed by down-sampling a 30-dimensional tempo feature by a factor of 3 and 2. Figure 7.2 shows an excerpt of Chopin's Mazurka Op. 68 No. 3 with a tempo change between two structural segments. We visualized the Fourier and autocorrelation tempograms together with the derived cyclic tempograms. One can see that the overall structure of the down-sampled cyclic tempograms is preserved.

Table 7.1 shows the averaged precision, recall and F-measure valuess computed on the

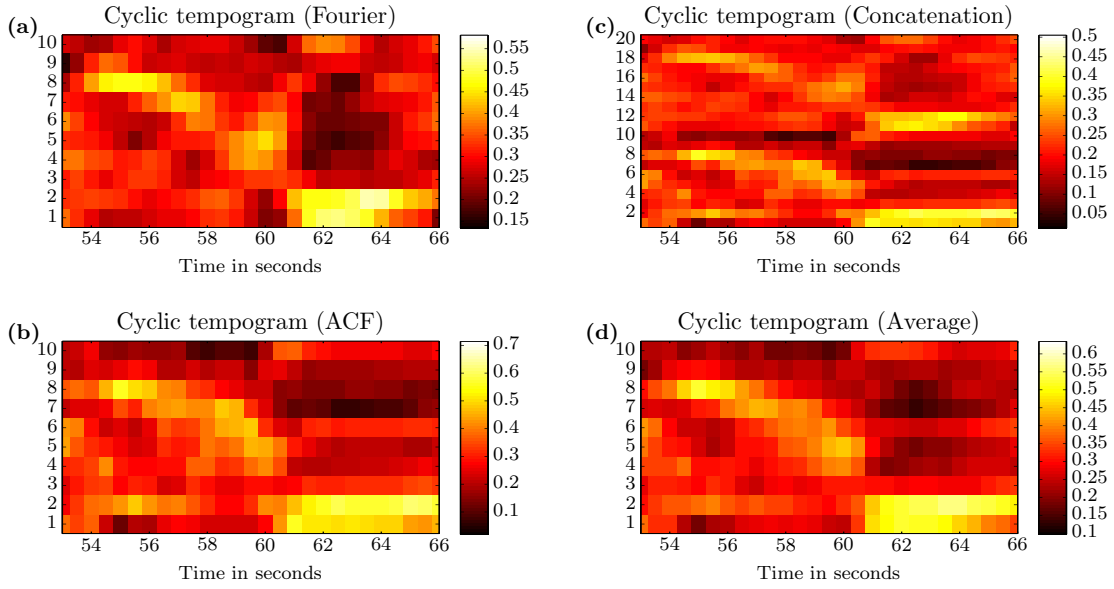


Figure 7.1. Cyclic Tempograms computed on the same excerpt as in Figure 7.2. All features are computed at a resolution of 2 Hz. **(a):** Cyclic tempogram computed from fourier tempogram. **(b):** Cyclic tempogram computed from autocorrelation tempogram. **(c):** Concatenation of (a) and (b). **(d):** Average of (a) and (b).

		ctF			ctACF		
d	L	P	R	F	P	R	F
10	17	0.24	**0.75**	0.34	0.26	0.68	0.35
	33	0.31	0.58	0.37	0.35	0.62	0.41
	49	0.36	0.52	**0.38**	**0.37**	0.51	0.40
15	17	0.24	0.75	0.34	0.25	0.68	0.35
	33	0.32	0.59	0.37	0.36	0.64	0.42
	49	0.36	0.51	0.38	0.37	0.51	0.40
30	17	0.24	0.74	0.34	0.25	**0.69**	0.35
	33	0.32	0.59	0.37	0.37	0.64	**0.43**
	49	**0.36**	0.51	0.38	0.37	0.51	0.40

Table 7.1. d: number of dimensions of feature, L: kernel length, Tolerance: ± 1 measure

		ctAvg			ctC		
d	L	P	R	F	P	R	F
10	17	0.25	**0.76**	0.35	0.25	**0.74**	0.35
	33	0.32	0.61	0.38	0.32	0.61	0.39
	49	**0.38**	0.51	**0.40**	**0.38**	0.50	**0.40**

Table 7.2. d: number of dimensions of feature, L: kernel length, Tolerance: ± 1 measure

whole Mazurka dataset.

We can conclude that the performance is not significantly changing between the different dimensions d. The only parameter which significantly affects the performance is the kernel length. If we fix a kernel length, then we have nearly the same performance with 10-, 15- and 30-dimensional features. Following the principle of Occam's razor [13] which basically states, that if two theories make the same predictions, the simpler one is better, we fix $d = 10$ for the next experiment.

We merge the two cyclic tempograms following two strategies. First, we compute their

average and second, we concatenate them, such that the resulting feature has twice as many dimensions as the original features. Figure 7.1 shows a visualization of the averaged feature ctAvg and the concatenated feature ctC together with the features from which they were derived.

Table 7.2 shows the results which do not yield any significant improvements. The performance values of the merged features are slightly higher than the ones of ctF and slightly lower than for ctACF. As the ctACF features showed the best performance, we will fix them with $d = 10$ as our tempo feature for further experiments.

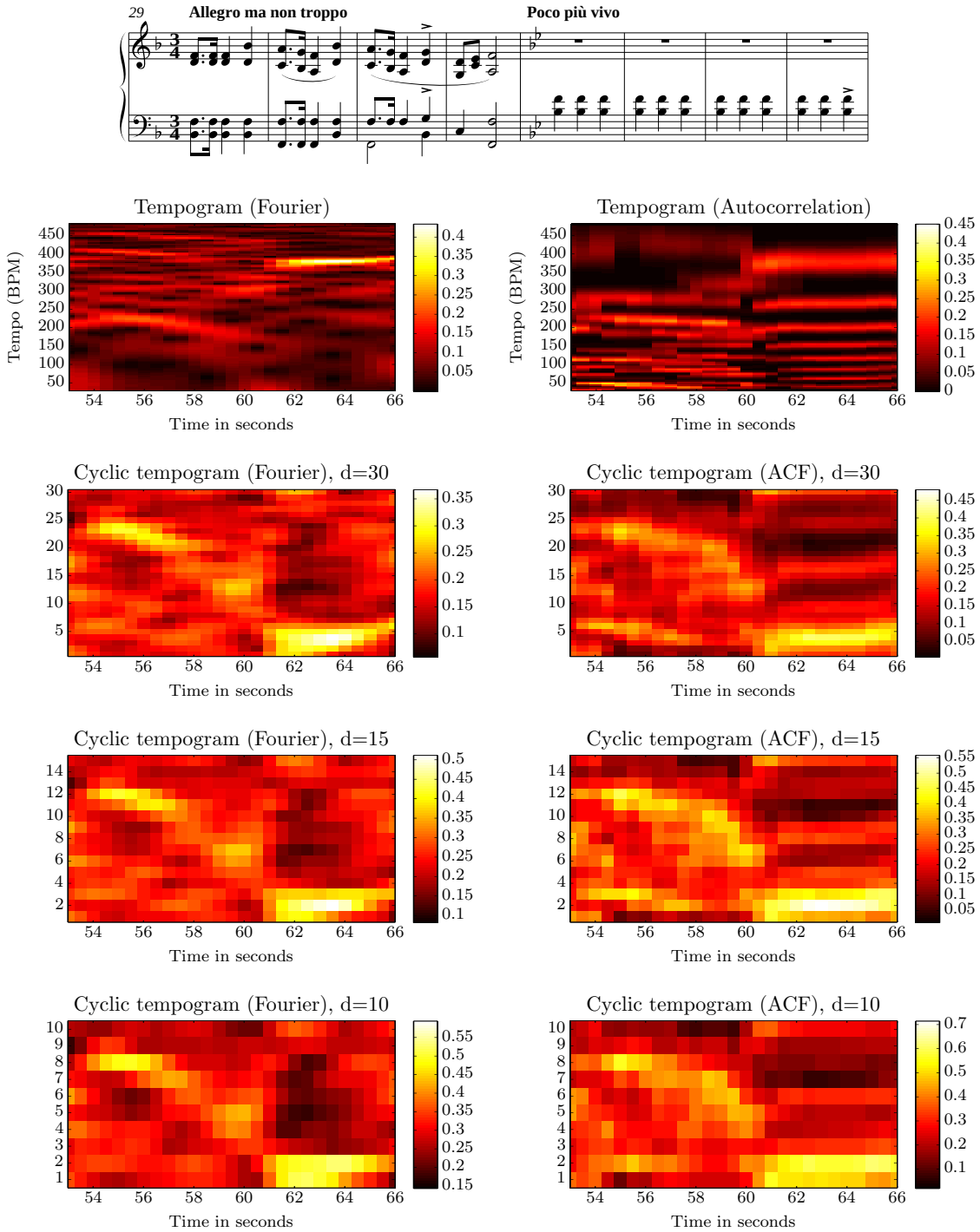


Figure 7.2. Top: Excerpt from Chopin Mazurka Op. 68 No. 3, measures 29-36, played by Bacha in 1997. **Left Column:** Fourier tempogram with a resolution of 2 Hz from which cyclic tempograms were derived on 3 granularity levels, $d = 30$, $d = 15$, $d = 10$. **Right Column:** Autocorrelation tempogram with a resolution of 2 Hz from which cyclic tempograms were derived on 3 granularity levels, $d = 30$, $d = 15$, $d = 10$.

7.2 Individual Interpretations vs Cross Version

In this section, we conduct an experiment to test our main hypothesis, that musically relevant points of novelty are consistent across different recordings. To this end, we compare the average performance of the boundary detection on the individual novelty curves to the performance of the boundary detection on the fusion curve. To compute the average performance values for the individual version, we apply the following procedure:

1. Compute the the boundaries for each interpretation individually on the physical time axis.
2. Warp the estimated boundaries to the MIDI reference.
3. Round boundaries to the next beat and compute measure position.
4. Calculate P, R, F for each version.
5. Average P, R, F over all interpretations of a piece.

The average performance values of the fusion novelty curves are computed as follows:

1. Compute fusion curve for each set of recordings.
2. Compute P, R, F for each fusion curve.
3. Calculate P, R, F.
4. Average the P, R, F values of all fusion curves.

Additionally, Table 7.3 shows the average variance in the novelty matrices, the average variance in the rough shape matrix and the MAD score which we introduced in Section 6.4.1.

	$\overline{\sigma^2(\mathcal{V})}$	$\overline{\sigma^2(RS)}$	MAD
CENS	4.99e-005	0.30	0.48
MFCC	1.45e-004	0.40	0.57
ctACF	1.96e-005	0.51	0.63

Table 7.3. $\overline{\sigma^2(\mathcal{V})}$: Average variance in all novelty matrices, $\overline{\sigma^2(RS)}$: variance of rough shape matrix, MAD: mean average deviation score.

The average P, R, F values are presented in Table 7.4. The full results of the individual version are shown in Table 7.5 and the full results for the cross-version approach are shown in Table 7.6.

We can see slight improvements of the performance values for CENS and MFCC features. The largest performance increase can be observed in the ctACF features. Note that for the ctACF feature, the recall increases from 0.46 to 0.62 and makes the main contribution

	CENS			MFCC			ctACF		
	P	R	F	P	R	F	P	R	F
Individual Versions	0.36	0.67	0.43	0.38	0.54	0.41	0.29	0.46	0.33
Cross-Version	0.40	0.68	0.47	0.44	0.64	0.48	0.35	0.62	0.41

Table 7.4. Average P, R, F values for boundary detection over all individual music recordings of the Mazurka dataset on CENS, MFCC, and ctACF features. First the average for all versions of a piece was computed and then the average over all pieces in the dataset was built. A kernel length of 33 and a tolerance of ± 1 measure was used.

to the performance increase. The ctACF also exhibit the highest $\overline{\sigma^2(RS)}$ and MAD value. The average variance $\overline{\sigma^2(\mathcal{V})}$ over the novelty matrices is not as robust as the other two measure, as it is heavily dependent on the amplitude of the novelty curves. Overall, we can see a slight correlation of the variance of the rough shapes of the novelty curves and the increase of the performance values in the cross-version approach. This is actually not too surprising. Consider a novelty matrix, where all novelty curves perfectly agree. In this case, we would not gain any additional information. The more variance there is in a set of recordings, the more information we can get out of it by combining the results on the individual recordings. The higher variance in the novelty curves of the ctACF features explain also the increase in the recall, as the average of these curves results in a novelty curve with more fluctuations and hence more peaks.

From this experiment we can conclude, that the cross version approach yields to a performance increase compared to the performance of the individual recordings. The results from this experiment support our hypothesis. But have to be cautious about these results. The test set consist only of 49 pieces. What is even more important, is that the dataset is not completely clean. There are sometimes versions of a piece in a set of recordings which belong to another piece. Furthermore, the reference annotations are still in a state of a first draft.

Performance values do not tell the whole story. Therefore, we will inspect some of the results in more depth in the next section.

Piece			CENS			MFCC			ctACF		
ID	M	B	P	R	F	P	R	F	P	R	F
06-1	112	7	0.25	0.78	0.38	0.35	**0.80**	0.48	0.26	0.65	0.37
06-2	96	9	0.39	0.85	0.53	0.37	0.63	0.46	0.34	0.59	0.42
06-3	98	12	0.42	0.45	0.43	0.69	0.62	0.65	0.27	0.26	0.26
06-4	40	9	0.73	0.42	0.52	0.41	0.16	0.22	0.41	0.21	0.27
07-1	104	9	0.67	0.90	0.76	0.38	0.60	0.46	0.27	0.43	0.33
07-2	118	14	0.30	0.60	0.39	0.27	0.46	0.33	0.28	0.45	0.34
07-3	105	11	0.44	0.74	0.54	0.47	0.46	0.45	0.33	0.39	0.35
07-4	60	7	**0.94**	0.91	**0.92**	0.62	0.50	0.54	0.55	0.56	**0.55**
07-5	20	12	0.93	0.43	0.58	0.73	*0.16*	0.25	**0.80**	*0.16*	0.26
17-1	100	10	0.53	0.78	0.63	0.50	0.54	0.51	0.25	0.38	0.30
17-2	68	3	0.14	0.75	0.23	0.17	0.72	0.28	0.19	**0.78**	0.31
17-3	168	10	0.25	0.90	0.38	0.20	0.49	0.28	0.22	0.60	0.32
17-4	132	10	0.14	0.48	0.21	0.23	0.64	0.33	0.18	0.50	0.26
24-1	96	10	0.34	0.65	0.44	0.26	0.46	0.32	0.23	0.42	0.29
24-2	120	16	0.60	0.51	0.55	0.70	0.53	0.60	0.52	0.40	0.45
24-3	79	6	0.32	0.63	0.42	0.28	0.52	0.36	0.26	0.48	0.33
24-4	186	20	0.31	0.53	0.39	0.31	0.50	0.38	0.34	0.49	0.39
30-1	53	4	0.26	0.96	0.41	0.29	0.69	0.40	0.27	0.65	0.38
30-2	64	7	0.37	0.55	0.43	0.42	0.34	0.36	0.36	0.41	0.38
30-3	111	10	0.32	0.72	0.44	0.34	0.63	0.44	0.21	0.36	0.26
30-4	139	14	0.32	0.66	0.43	0.35	0.67	0.46	0.31	0.54	0.39
33-1	48	4	0.33	**0.98**	0.49	0.25	0.70	0.36	0.25	0.65	0.36
33-2	143	16	0.71	0.48	0.57	0.72	0.67	0.69	0.30	0.26	0.27
33-3	48	3	0.18	0.70	0.28	0.23	0.58	0.32	0.16	0.64	0.25
33-4	224	19	0.30	0.55	0.38	0.31	0.57	0.40	0.23	0.38	0.28
41-1	139	14	0.42	0.70	0.52	0.39	0.54	0.45	0.29	0.42	0.34
41-2	68	7	0.34	0.89	0.49	0.28	0.54	0.36	0.29	0.61	0.39
41-3	78	13	0.49	*0.35*	0.40	0.64	0.30	0.40	0.44	0.22	0.29
41-4	74	8	0.31	0.65	0.41	0.32	0.46	0.37	0.25	0.39	0.30
50-1	104	6	0.28	0.88	0.42	0.26	0.47	0.33	0.18	0.42	0.25
50-2	127	10	0.24	0.59	0.34	0.24	0.43	0.30	0.27	0.51	0.35
50-3	208	10	0.24	0.86	0.38	0.22	0.66	0.33	0.19	0.54	0.28
56-1	204	13	0.29	0.73	0.41	0.32	0.64	0.43	0.18	0.35	0.23
56-2	92	8	0.56	0.61	0.57	0.62	0.51	0.53	0.38	0.39	0.37
56-3	220	15	0.25	0.74	0.37	0.25	0.62	0.35	0.20	0.49	0.29
59-1	142	9	0.16	0.56	0.25	0.16	0.45	0.23	0.17	0.44	0.24
59-2	111	4	0.21	0.87	0.33	0.14	0.49	0.21	*0.10*	0.38	*0.15*
59-3	154	11	0.23	0.59	0.33	0.27	0.45	0.33	0.18	0.35	0.24
63-1	102	9	0.27	0.53	0.36	0.48	0.58	0.52	0.29	0.37	0.32
63-2	56	4	0.18	0.65	0.28	0.25	0.70	0.37	0.21	0.58	0.31
63-3	76	8	0.25	0.50	0.33	0.38	0.61	0.46	0.31	0.50	0.38
67-1	60	7	0.52	0.59	0.55	0.57	0.38	0.43	0.28	0.29	0.28
67-2	72	6	0.23	0.65	0.34	0.32	0.62	0.41	0.24	0.49	0.32
67-3	56	3	0.18	0.70	0.29	0.38	0.74	0.49	0.22	0.62	0.32
67-4	112	6	*0.10*	0.42	*0.16*	*0.12*	0.39	*0.17*	0.20	0.59	0.30
68-1	84	11	0.55	0.66	0.59	0.59	0.49	0.53	0.47	0.45	0.45
68-2	84	11	0.51	0.90	0.64	0.46	0.72	0.55	0.42	0.71	0.52
68-3	60	9	0.35	0.50	0.41	**0.76**	0.65	**0.69**	0.44	0.51	0.47
68-4	63	7	0.23	0.64	0.34	0.25	0.52	0.32	0.23	0.51	0.31
Ø			0.36	0.67	0.43	0.38	0.54	0.41	0.29	0.46	0.33

Table 7.5. Average P, R, F values of the novelty detection applied to each version separately. Kernel: $L = 33$, Tolerance: 1 measure.

Piece			CENS			MFCC			ctACF		
ID	M	B	P	R	F	P	R	F	P	R	F
06-1	112	7	0.24	0.86	0.38	0.35	**1.00**	0.52	0.37	**1.00**	0.54
06-2	96	9	0.45	**1.00**	0.62	0.46	0.67	0.55	0.47	**1.00**	0.64
06-3	98	12	0.50	0.50	0.50	0.91	0.83	0.87	0.21	0.25	0.23
06-4	40	9	**1.00**	0.44	0.62	*0.00*	*0.00*	*0.00*	0.33	0.22	0.27
07-1	104	9	0.82	**1.00**	0.90	0.50	0.89	0.64	0.35	0.67	0.46
07-2	118	14	0.29	0.50	0.37	0.28	0.50	0.36	0.34	0.71	0.47
07-3	105	11	0.45	0.82	0.58	0.70	0.64	0.67	0.50	0.45	0.48
07-4	60	7	**1.00**	**1.00**	**1.00**	0.71	0.71	0.71	0.78	**1.00**	**0.88**
07-5	20	12	**1.00**	0.50	0.67	**1.00**	0.17	0.29	**1.00**	*0.08*	0.15
17-1	100	10	0.46	0.60	0.52	0.43	0.60	0.50	*0.11*	0.20	*0.14*
17-2	68	3	0.17	0.67	0.27	0.18	0.67	0.29	0.33	**1.00**	0.50
17-3	168	10	0.27	**1.00**	0.43	0.24	0.70	0.36	0.32	**1.00**	0.49
17-4	132	10	*0.06*	*0.20*	*0.10*	0.17	0.60	0.27	0.12	0.40	0.19
24-1	96	10	0.38	0.60	0.46	0.22	0.50	0.30	0.32	0.70	0.44
24-2	120	16	0.69	0.56	0.62	0.83	0.63	0.71	0.67	0.63	0.65
24-3	79	6	0.33	0.67	0.44	0.40	0.33	0.36	0.40	0.67	0.50
24-4	186	20	0.35	0.65	0.46	0.38	0.75	0.51	0.46	0.90	0.61
30-1	53	4	0.29	**1.00**	0.44	0.25	0.50	0.33	0.38	0.75	0.50
30-2	64	7	0.40	0.57	0.47	0.75	0.43	0.55	0.50	0.71	0.59
30-3	111	10	0.32	0.80	0.46	0.36	0.80	0.50	0.17	0.30	0.21
30-4	139	14	0.35	0.64	0.45	0.31	0.64	0.42	0.32	0.86	0.47
33-1	48	4	0.40	**1.00**	0.57	0.30	0.75	0.43	0.36	**1.00**	0.53
33-2	143	16	0.80	0.50	0.62	0.94	0.94	0.94	0.27	0.25	0.26
33-3	48	3	0.29	0.67	0.40	0.13	0.33	0.18	0.22	0.67	0.33
33-4	224	19	0.24	0.47	0.32	0.26	0.58	0.35	*0.11*	0.21	0.15
41-1	139	14	0.46	0.79	0.58	0.56	0.71	0.63	0.33	0.50	0.40
41-2	68	7	0.35	0.86	0.50	0.31	0.71	0.43	0.26	0.71	0.38
41-3	78	13	0.44	0.31	0.36	0.83	0.38	0.53	0.50	0.23	0.32
41-4	74	8	0.28	0.63	0.38	0.42	0.63	0.50	0.33	0.50	0.40
50-1	104	6	0.29	**1.00**	0.44	0.29	0.67	0.40	0.22	0.67	0.33
50-2	127	10	0.25	0.60	0.35	0.19	0.40	0.26	0.35	0.80	0.48
50-3	208	10	0.24	0.90	0.38	0.31	0.80	0.44	0.31	0.80	0.44
56-1	204	13	0.32	0.85	0.47	0.33	0.62	0.43	0.12	0.31	0.17
56-2	92	8	0.83	0.63	0.71	0.86	0.75	0.80	0.63	0.63	0.63
56-3	220	15	0.24	0.73	0.37	0.24	0.67	0.35	0.24	0.67	0.35
59-1	142	9	0.13	0.44	0.20	0.14	0.56	0.22	0.19	0.56	0.28
59-2	111	4	0.27	**1.00**	0.42	0.14	0.50	0.22	0.17	0.75	0.27
59-3	154	11	0.25	0.64	0.36	0.33	0.64	0.44	0.20	0.36	0.26
63-1	102	9	0.24	0.44	0.31	0.54	0.78	0.64	0.46	0.67	0.55
63-2	56	4	0.17	0.50	0.25	0.27	**1.00**	0.42	0.25	0.75	0.38
63-3	76	8	0.29	0.50	0.36	0.43	0.75	0.55	0.40	0.75	0.52
67-1	60	7	0.57	0.57	0.57	0.83	0.71	0.77	0.14	0.14	*0.14*
67-2	72	6	0.20	0.50	0.29	0.40	**1.00**	0.57	0.27	0.50	0.35
67-3	56	3	0.20	0.67	0.31	0.43	**1.00**	0.60	0.25	**1.00**	0.40
67-4	112	6	0.09	0.50	0.16	0.09	0.33	0.14	0.24	0.83	0.37
68-1	84	11	0.83	0.91	0.87	0.56	0.45	0.50	0.60	0.55	0.57
68-2	84	11	0.69	**1.00**	0.81	0.56	0.91	0.69	0.58	**1.00**	0.73
68-3	60	9	0.31	0.44	0.36	**1.00**	0.89	**0.94**	0.45	0.56	0.50
68-4	63	7	0.27	0.86	0.41	0.25	0.43	0.32	0.20	0.71	0.31
Ø			0.40	0.68	0.47	0.44	0.64	0.48	0.35	0.62	0.41

Table 7.6. P, R, F values of the novelty detection on the average novelty curve $\mathring{\mathcal{N}}_{avg}$, Kernel: $L = 33$, Tolerance: 1 measure.

7.3 Discussion of Individual Results

In the following the abbreviation "mm." stands for measure.

7.3.1 Mazurka Op. 68 No. 3

In Figure 7.3, we show an excerpt of the boundary detection results corresponding to mm. 31-39 of Chopin's Mazurka Op. 68 No. 3. Figure 7.3a shows the corresponding excerpt of the musical score. There are two important changes in musical aspects which are clearly indicated in the score. First, there is a change of tonality from F major to $B^\flat$ major in mm. 33. Second, again in mm. 33, there is a change of tempo indicated by "Poco più vivo" (a little bit more lively). The tempo in the part before is indicated as "Allegro ma non troppo" which means "fast, but not too much". As the performance instruction in mm. 33 is to play more lively, this means to play faster than before.

Figure 7.3b shows the results computed on CENS features. The change of tonality is perfectly reflected as it is exactly detected at the start of mm. 33. In the annotation, there is another boundary at mm. 37 which indicates the start of a phrase. As there is no change in the coarse harmonic structure, the novelty curve on the CENS features does not detect a boundary.

The results of the boundary detection with MFCC features is shown in Figure 7.3c. The boundary at mm. 33 is recovered, but only within the tolerance region. What the MFCCs probably capture here is a change in the sound intensity, as most performers tend to perform "decrescendo" (incremental decrease in loudness) at this position in the piece. The second boundary is detected only one beat position too early. At the next beat position, the right hand joins in (upper voice), two octaves higher than the left hand. Arguably the piano has a different timbre if one compares the lower octaves to the higher ones. Thus, the MFCC captured probably a change in timbre at this point.

Figure 7.3d depicts the results for the tempo feature. Both boundaries are recovered within the tolerance region. The first boundary at mm. 33 is detected one beat before mm. 33. Just before the tempo change, most of the performers play a short ritardando (incremental decrease in tempo), hence it is not too surprising that the boundary is detected before the annotated boundary, as the tempo changes already in advance. The second boundary is detected one measure too late. Around the annotated boundary, most performers start to play faster, but most of them do this after mm. 37.

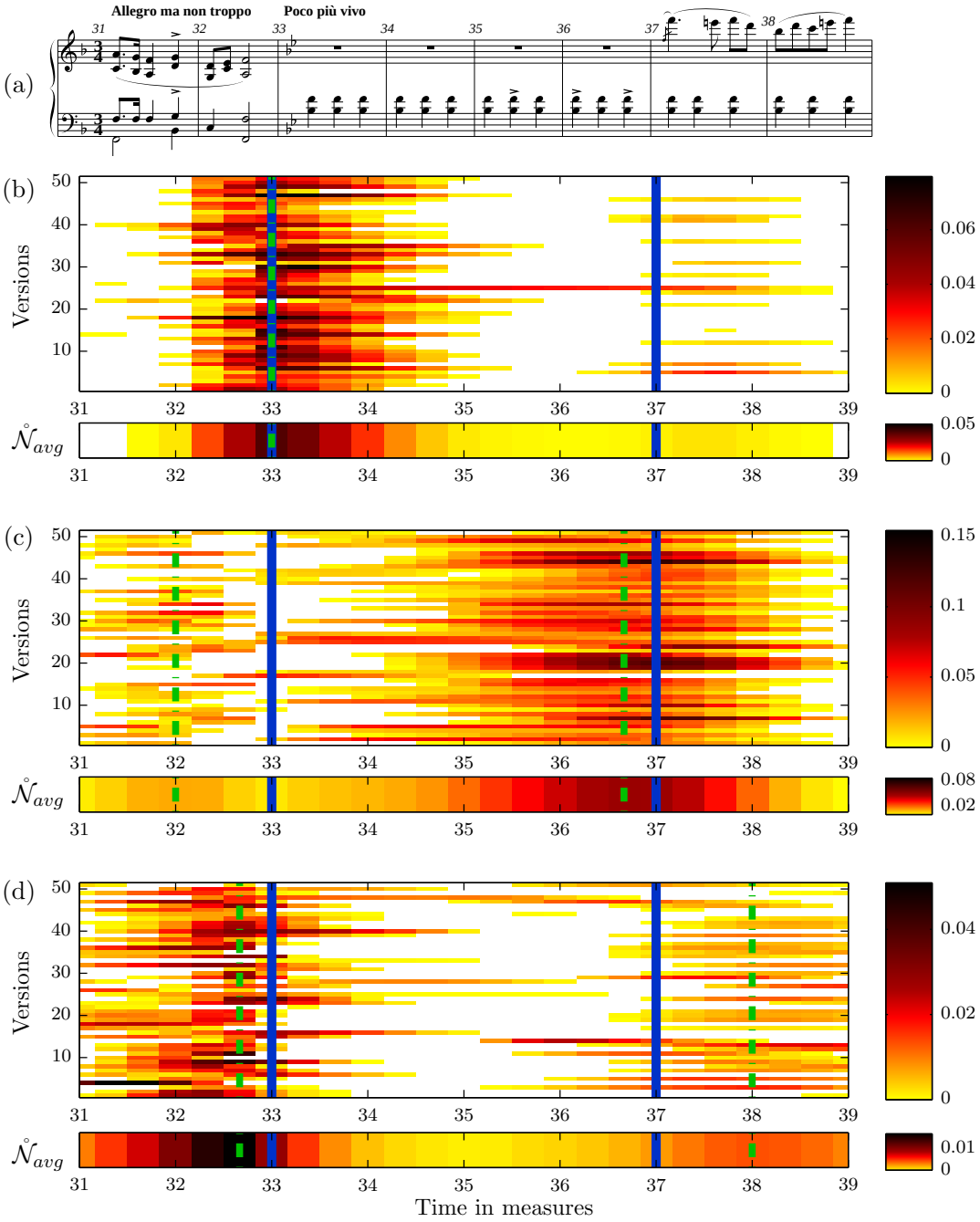


Figure 7.3. Excerpt of the results of the boundary detection on the fusion curves $\mathring{\mathcal{N}}_{avg}$ on Chopin's Mazurka Op. 68 No. 3 mm. 31-39. A kernel with $L = 33$ and a tolerance of one measure was used. The dashed green lines mark boundaries classified as correct, and the blue lines mark the position of the ground truth boundaries. **(a):** Excerpt from score. **(b):** Top: Excerpt of Novelty Matrix extracted from CENS features, Bottom: Excerpt of the corresponding average novelty curve $\mathring{\mathcal{N}}_{avg}$. **(c):** Top: Excerpt of Novelty Matrix extracted from MFCC features, Bottom: Excerpt of the corresponding average novelty curve $\mathring{\mathcal{N}}_{avg}$. **(d):** Top: Excerpt of Novelty Matrix extracted from ctACF features, Bottom: Excerpt of the corresponding average novelty curve $\mathring{\mathcal{N}}_{avg}$.

7.3.2 Mazurka Op. 63 No. 3

Figure 7.4 shows an example where all features exhibit a boundary around mm. 13 which is not in the annotations. This example is of particular interest as the boundary can be explained musically. In mm. 13 starts a short phrase.

The novelty detection on the CENS features assigns a boundary to the second beat position of mm. 12, see Figure 7.4b. This Mazurka has the tonality $C^\sharp$ minor. What the CENS features actually capture here is a modulation into the tonality E major, which is the relative major key to $C^\sharp$. The modulation lasts for four measures before the tonality is changed again. This time, the novelty detection also detects a boundary which is present in the reference annotation as well.

The novelty detection on the MFCC features assigns a boundary at the second beat of mm. 13, see Figure 7.4c. This is exactly the position where the right hand (upper voice) plays in a higher octave. Hence, we have a timbre change here.

The tempo features show a peak at the first beat of mm. 13, see Figure 7.4d. Most performers slow down at this passage to emphasize the phrase which starts at mm. 13.

This example shows that even a novelty peak which is classified as wrong does not necessarily lack musical meaning. Here, all the fusion novelty curves captured consistently a musically important change which is not reflected in the reference annotation, however.

7.4 Dataset Overview

In Appendix B, the novelty matrices for all pieces in the dataset are shown together with the corresponding fusion curves and the results of the boundary detection.

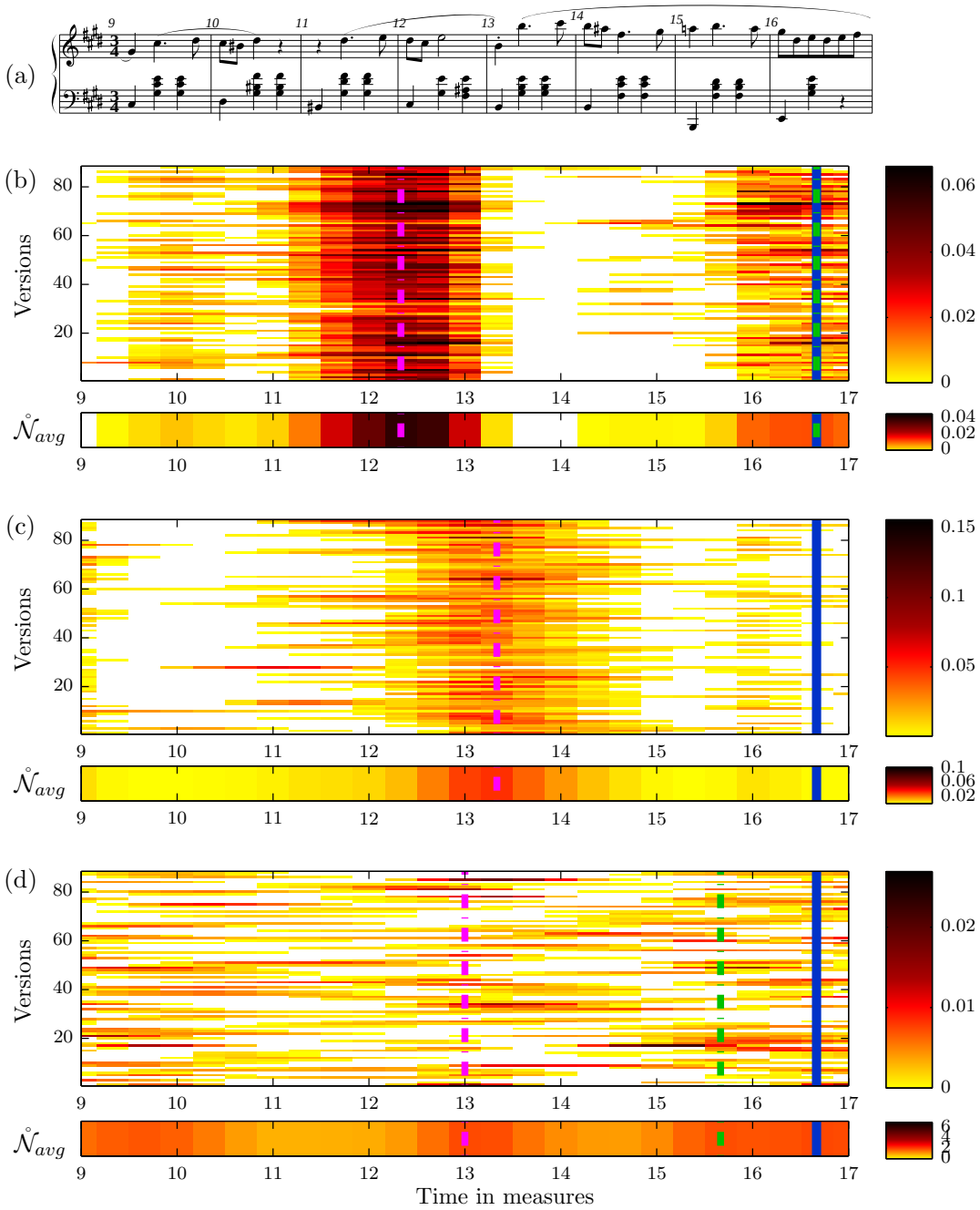


Figure 7.4. Excerpt of the results of the boundary detection on the fusion curves $\mathring{\mathcal{N}}_{avg}$ on Chopin's Mazurka Op. 63 No. 3 mm. 9-16. A kernel with $L = 33$ and a tolerance of one measure was used. The dashed green lines mark boundaries classified as correct, the magenta lines mark boundaries which are classified as wrong and the blue lines mark the position of the ground truth boundaries. **(a):** Excerpt from score. **(b):** Top: Excerpt of Novelty Matrix extracted from CENS features, Bottom: Excerpt of the corresponding average novelty curve $\mathring{\mathcal{N}}_{avg}$. **(c):** Top: Excerpt of Novelty Matrix extracted from MFCC features, Bottom: Excerpt of the corresponding average novelty curve $\mathring{\mathcal{N}}_{avg}$. **(d):** Top: Excerpt of Novelty Matrix extracted from ctACF features, Bottom: Excerpt of the corresponding average novelty curve $\mathring{\mathcal{N}}_{avg}$.

"Simplicity is the highest goal, achievable when you have overcome all difficulties. After one has played a vast quantity of notes and more notes, it is simplicity that emerges as the crowning reward of art."

Frédéric Chopin

8 Conclusions

In this thesis, we presented a pipeline for the cross-version analysis of novelty detection algorithms. We applied the pipeline to a well-known algorithm for the extraction of novelty curves from self-distance matrices, which are computed from feature sequences. We used three different audio features to account for the musical aspects harmony, timbre, and tempo. The resulting curves were then transformed onto the musical time axis and adjusted to the musical beat positions by linear interpolation. We have shown how this approach can be used to combine the individual novelty curves from different versions into a fusion curve.

Furthermore, we conducted experiments showing some evidence for our hypothesis that musically important points of novelty are consistent across different versions of a piece. We gained some increase in precision, recall and F-measure by applying the novelty detection on the fusion curve. Nevertheless, we have to be cautious about these results, as these experiments were limited to a small dataset and only slight improvements of the performance values were seen. Furthermore, the reference annotations were still in a draft condition: There were some inconsistencies in the granularity of the annotations. Some segments were annotated on phrase level and some on section level. As performance values on their own do not suffice to assess the results of an algorithm, we showed some results in detail and discussed their musical relevance. We have seen that musically meaningful outcomes are not always reflected in the reference annotation.

Apart from looking at the consistency of points of novelty, the main contribution of the cross-version approach is that it provides an analysis framework. We showed a visualization which helps to analyze the results of an algorithm and to get an overview of a dataset (see Appendix B). The main advantage of the cross-version approach is that all results from individual versions of a piece of music can be visualized on a common time axis. We used the time axis of a MIDI reference version from which we also extracted the beat positions. This enabled us to use a common musical time axis. This has furthermore the advantage

that the results can easily be compared to a musical score. That way, it is easier for a musically skilled person to connect the results with the musical reasons provided by the score.

In the following, we will point out improvements and future work. We will first address some general limitations that have been set in this thesis and their implications for future work. Secondly, we address some problems that have been identified during this research and propose strategies to overcome them. Lastly, we will point out some future research directions that have not been addressed in this work.

The feature set used in this thesis was restricted to a set of three audio features. These features were used to account for the musical aspects harmony, timbre, and tempo in music recordings. However, music consists of many more dimensions and there are various other audio features which could be incorporated into our approach. Some examples for such features are loudness, rhythm, tonality or melody. From a signal processing perspective, the feature easiest to include would be loudness. Other musical aspects such as melody are hard to extract even from symbolic representations. Concerning the feature resolution, one major drawback in our pipeline is that we used a constant feature rate. We discussed the problems which arise from this decision in different parts of this thesis. In future work, a tempo adaptive feature extraction method would be a promising strategy. This could be implemented by exploiting meta data from a MIDI version or by using a beat tracking algorithm.

We evaluated our approach on a set of Mazurkas, which were written by Chopin in a specific style and are all performed on piano. The choice of this dataset was based on the large number of available recordings for each piece of music. Future research should include a more diverse set of compositional styles and music with different instrumentation. Furthermore, the cross-version approach is not necessarily restricted to classical music. Classical music was chosen because there often exist many recordings for the same piece of music. If different versions of a piece are not available, one single digital version could be sufficient. One could for example generate many audio versions from a MIDI version, where parameters such as tempo, loudness, instrumentation or key can easily be changed. With these synthetic versions, the robustness of an algorithm could be tested on a large dataset with a high variety of musical parameters. Furthermore, this approach could help to identify problematic musical settings where the tested algorithm fails.

Another limitation is that only one algorithm for novelty detection was tested in the context of this research. On the basis of this work, different algorithms can be compared within our approach, and strategies to combine different algorithms could be evaluated.

Furthermore, we did only test one method to compute a fusion result, the arithmetic mean of all novelty curves from the same feature. Another approach would be to use the rough shape of the novelty curves to measure the agreement at a specific position. Then, a majority vote could be done so that the computation of the fusion curve is based on those curves exclusively which agree at that position.

To conclude, there are many levels in the pipeline on which one could test different approaches:

1. Level 1 - Features

2. Level 2 - Self-Distance Matrix
3. Level 3 - Individual Novelty Curves
4. Level 4 - Fusion (Average) Novelty Curves
5. Level 5 - Peaks of Novelty Curves

On level 1, we performed an experiment to compute a fusion feature from our two tempogram features. To this end, we applied two simple methods to combine the features: averaging and concatenation. On level 2, all self distance matrices could be resampled to a common resolution and combined by averaging. On level 3, the individual novelty curves from different novelty detection algorithms could be combined. Or, even simpler, the novelty curves which resulted from different features could be combined on this level. On level 4, the results from different fusion curves could be combined. On level 5, the final peaks resulting from different novelty curves could be merged into a common result, e.g. by a majority vote or a weighting scheme.

Last but not least, the cross-version approach could bridge the gap between the analysis of audio recordings and the analysis of music in symbolic formats. In this framework it would be convenient to compare the results from these different domains, or even to combine them.

A

Meta MIDI Annotation Format

In this section we introduce the MetaMIDI annotation format. An example annotation is shown in Figure A.1. Note that it is possible to comment an annotation file with the percent ('%') symbol. Everything after the percent symbol is discarded by the parser.

```
%% HEADER
midiOffsetInQuarters 0;
upbeatInQuarters 0;
timeSignature 3/4;
quartersPerBeat 1;
numberOfMeasures 60;


%% STRUCTURE ANNOTATION
structure 001.000 008.999 "A1"
structure 009.000 016.999 "A2"
structure 017.000 020.999 "B1"
structure 021.000 024.999 "B2"
structure 025.000 032.999 "A3"
structure 033.000 036.999 "C"
structure 037.000 040.999 "D1"
structure 041.000 044.999 "D2"
structure 045.000 052.999 "A4"
structure 053.000 060.999 "A2"
```

Figure A.1. Example of annotation of *Chopin Mazurka Op68 No.3* in Meta MIDI format.

A.1 Keywords

upBeatInQuarters *number*
: Defines how long the upbeat is in quarter notes.

midiOffsetInQuarters *number*
: If there are rests inserted in the midi, which are not notated in the score then the field defines how many rests/beats (in quarter notes) are inserted before the first midi note event.

numberOfMeasures *number*
: Number of measures in the piece.

timeSignature *nominator/denominator*
: Global time signature according to the reference score

quartersPerBeat *number*
: The length of a musical beat in quarter notes.

timeSignatureChange *measure nominator/denominator*
: Optional field to annotate a time signature change. The measure field refers to the measure in the score

quartersPerBeatChange *measure*
: Optional field to change the length of a musical beat within a piece of music.

structure *start_measure end_measure label*
: In the case of an upbeat in the beginning of the piece, we theoretically fill the upbeat measure to a complete measure (measure 0) such that the starting position of a structure segment is associated with the theoretical position in a complete measure.

infoRepetition *start_i end_i label*

A.2 Toolbox

Reading Annotation Files

The `metaMIDI` function is used to parse a meta MIDI annotation file into a struct object.

Sample usage:

```
|
```

$$metaMidiAnnotation$$

```
= metaMIDI('mazurka07-4.meta');—

function [ metaMidiAnnotation ] = metaMIDI( filename )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Programmer: Thomas Praetzlich
% Description:
```

```
%
%   Reads a meta MIDI annotation file.
%
%
% Input:
%         filename
%
% Output:
%
%         metaMidiAnnotation.midiOffsetInQuarters
%         metaMidiAnnotation.upbeatInQuarters
%         metaMidiAnnotation.structure
%         metaMidiAnnotation.infoRepetition
%         metaMidiAnnotation.timeSignature
%         metaMidiAnnotation.timeSignatureChange
%         metaMidiAnnotation.quartersPerBeat
%         metaMidiAnnotation.quartersPerBeatChange
%         metaMidiAnnotation.numberOfMeasures
%         metaMidiAnnotation.lastMeasureInQuarters
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Extracting Musical Beat Positions from MIDI

The `getMusicalBeatPositionsInSec` function extracts beat positions from a MIDI file. It can either use the information stored in an annotation file to compute the musical beat, or extract a constant beat.

Sample usage:

```
[ metaMidiAnnotation ] = getMusicalBeatPositionsInSec(metaMidiAnnotation, tempoList, info, ...
                                                                          lengthMidi_sec,parameter)


function beatPositions_sec = getMusicalBeatPositionsInSec(metaMidiAnnotation, tempoList, info, ...
                                                                          lengthMidi_sec,parameter)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: getBeatPositionsInSec
% Version: 1
% Date: 2011
% Programmer: Thomas Praetzlich, Sebastian Ewert
%
% Description:
%   Computes the musical beat positions in seconds. The function can use
%   musical beats or a constant beat (e.g. quarters)
%
% Input:
%   parameter.computeMusicalBeat: compute the musical beat using
%              (default=1)        information from meta MIDI annotations
%
%
%   parameter.quartersPerBeat   : This option is only used if you turn off
%                                 parameter.computeMusicalBeat. It defines
```

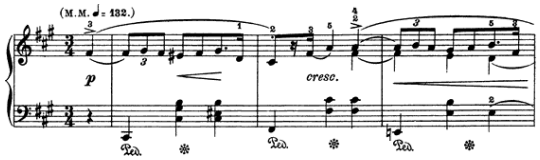
```
%                                  the number of quarter notes per beat
%              (default=1)         e.g.  quarter note:   1,
%                                        eighth note:    0.5,
%                                        dotted quarter: 1.5;...
%
%   metaMidiAnnotation         : structure containing meta information, as
%                                produced by metaMIDI
%
%   tempoList                  : as produced by midird4
%   info                       : as produced by midird4
%   lengthMidi_sec             : for example via  get_midilength_ms(midi)/1000
%
% Output:
%        beatPositions_sec
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

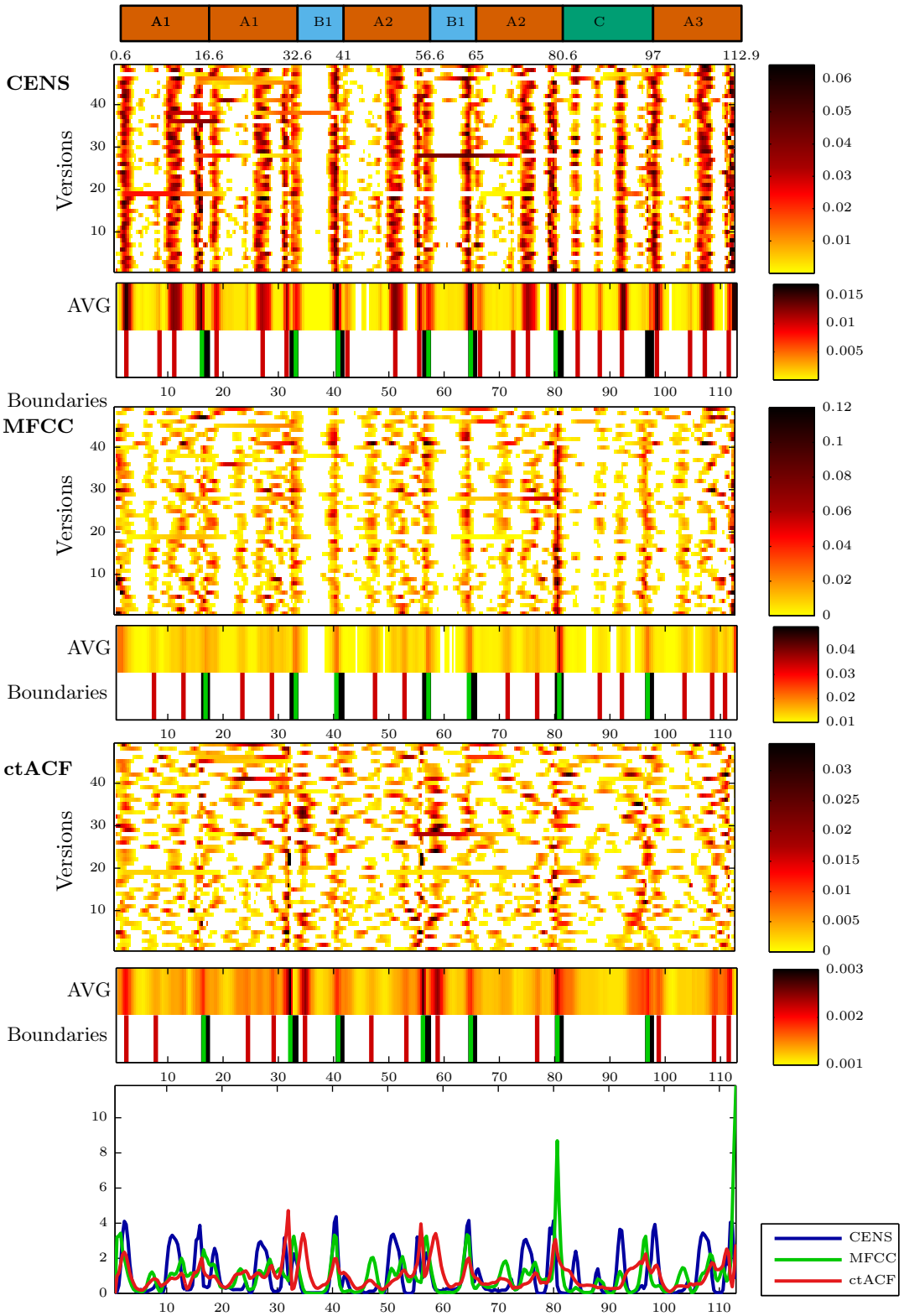
B

Dataset Overview

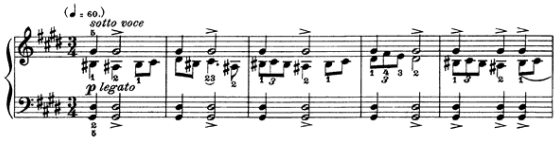
In this chapter, we show an overview of the results on the Mazurka dataset. For each piece in the dataset, a musical incipit and a reference annotation is shown. We show the novelty matrices for CENS, MFCC (4-13th coefficient) and ctACF features. Below each novelty matrix, the corresponding fusion curve (average novelty curve) is shown together with the results of the peak picking. The peak picking results are encoded in the following way: a red line marks a boundary which is not reflected in the reference, a green line marks a correct boundary and the black lines indicate the reference annotations. At the bottom of each page, all fusion curves are depicted in the same line plot. Note that these curves have been normalized by their average values for the purpose of better visibility in the plot.

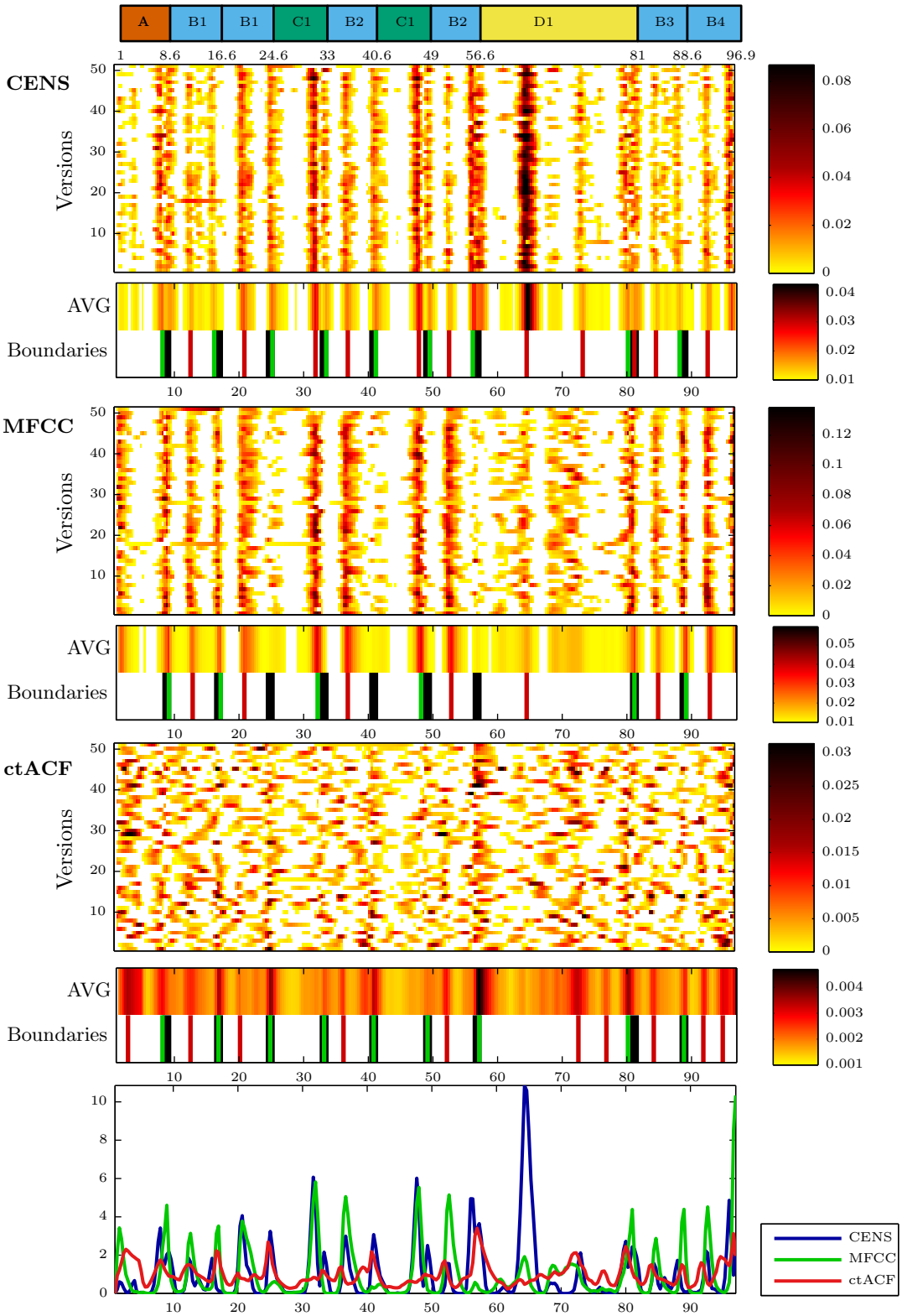
Mazurka 6 No. 1



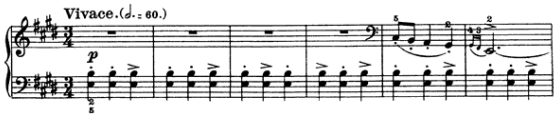


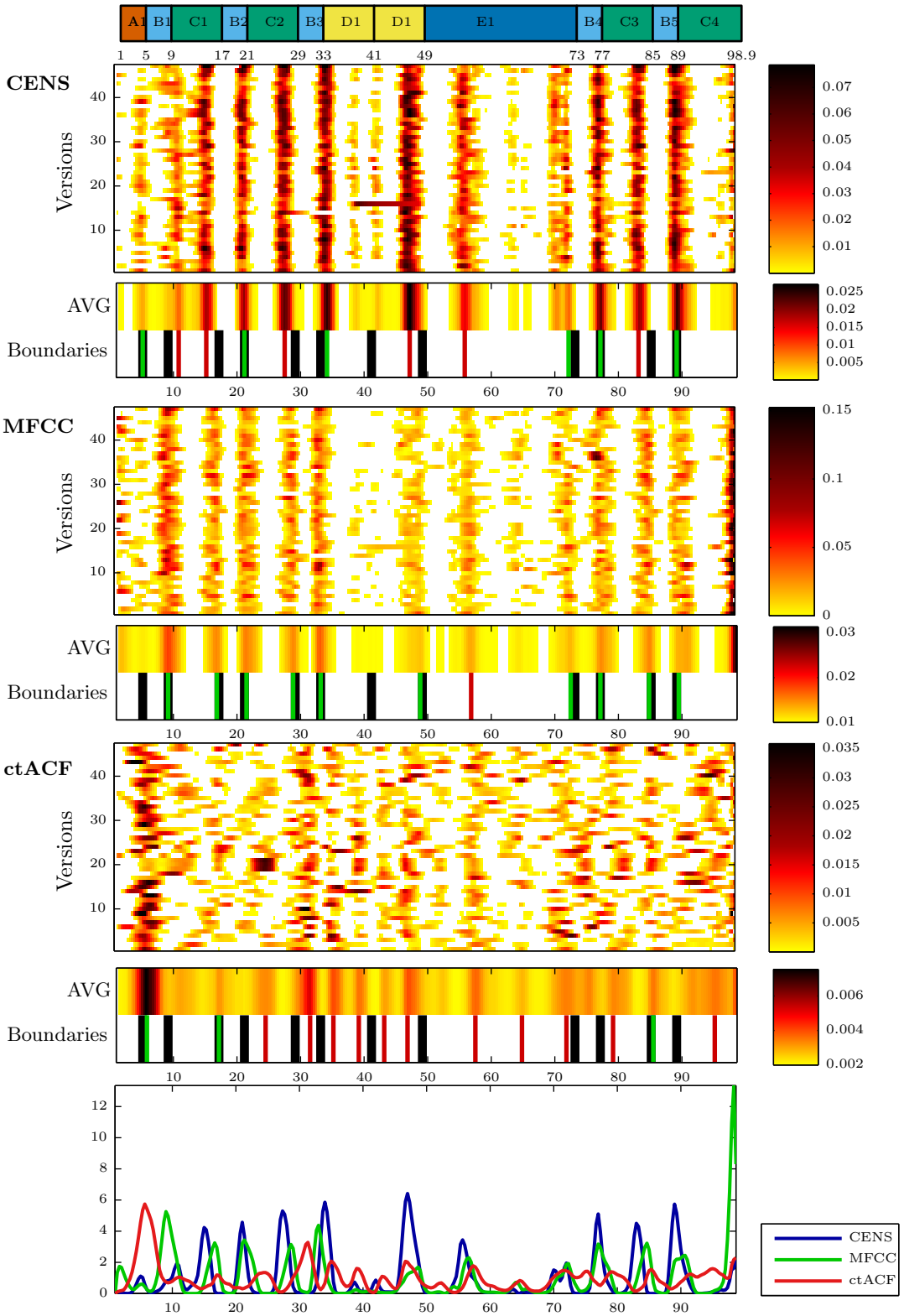
Mazurka 6 No. 2



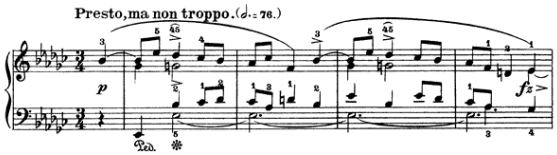


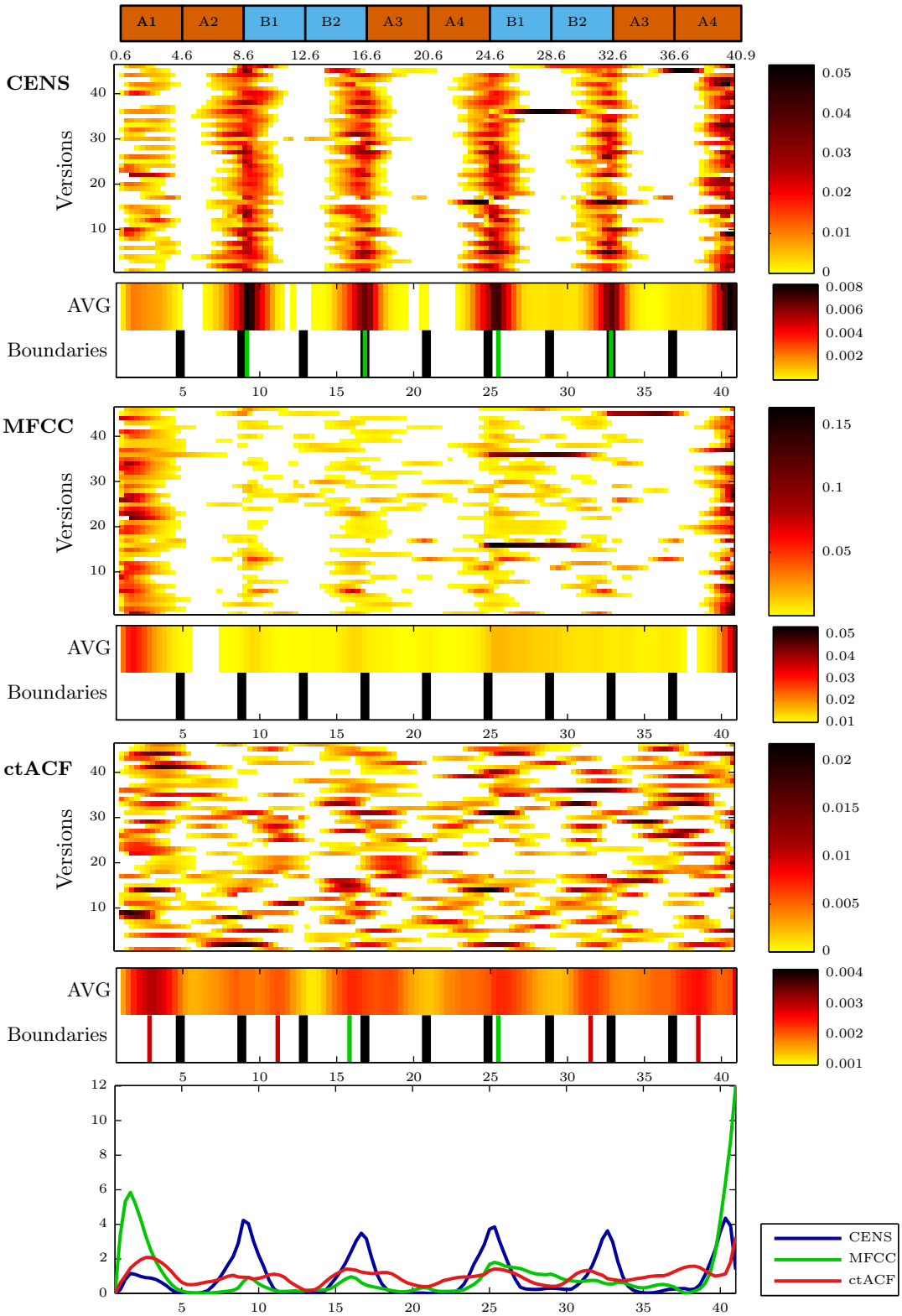
Mazurka 6 No. 3



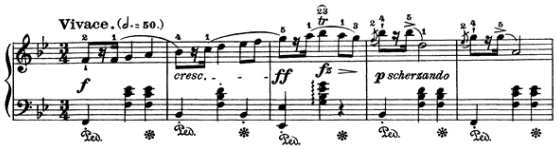


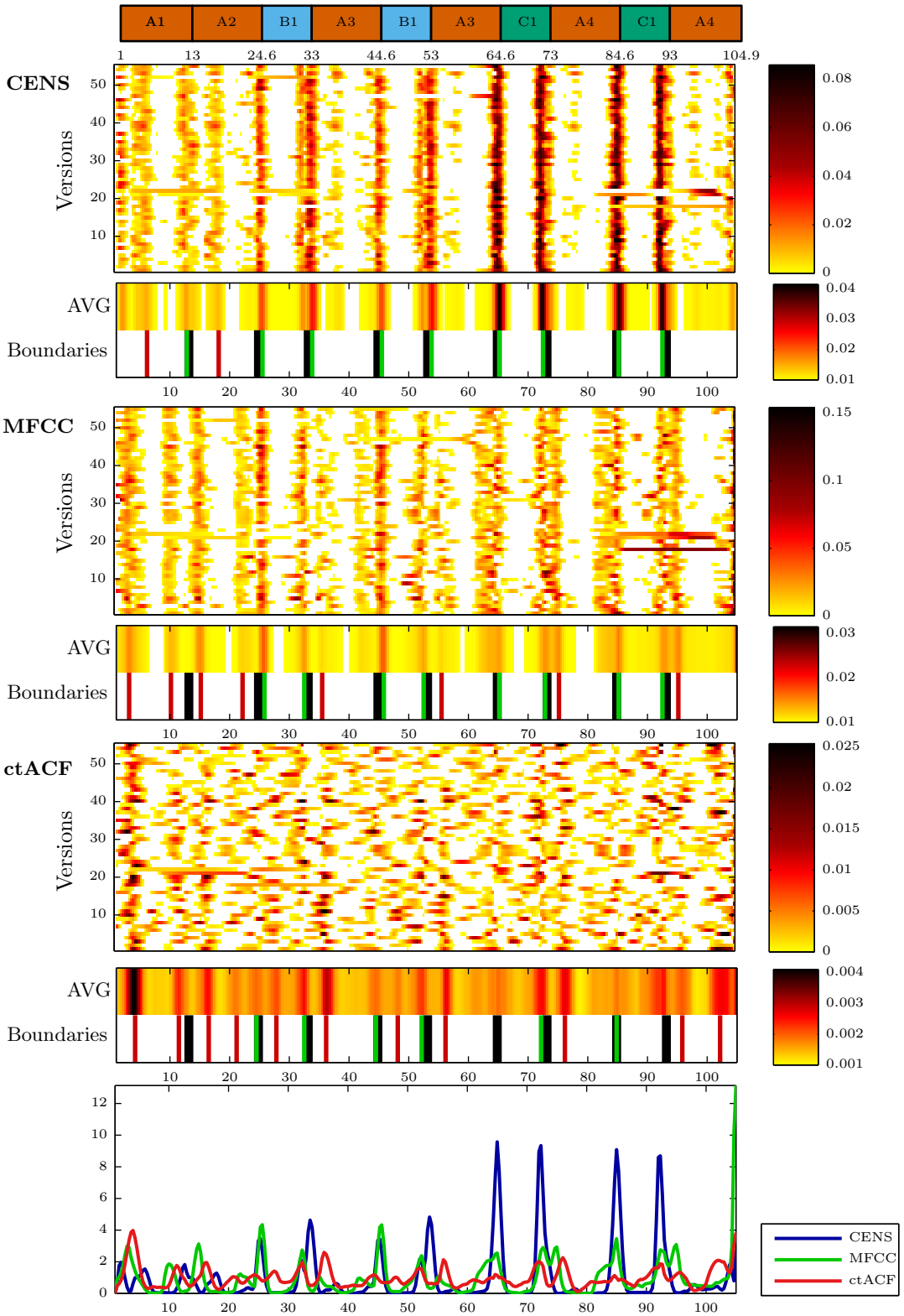
Mazurka 6 No. 4



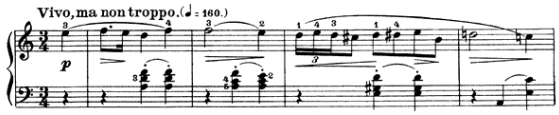


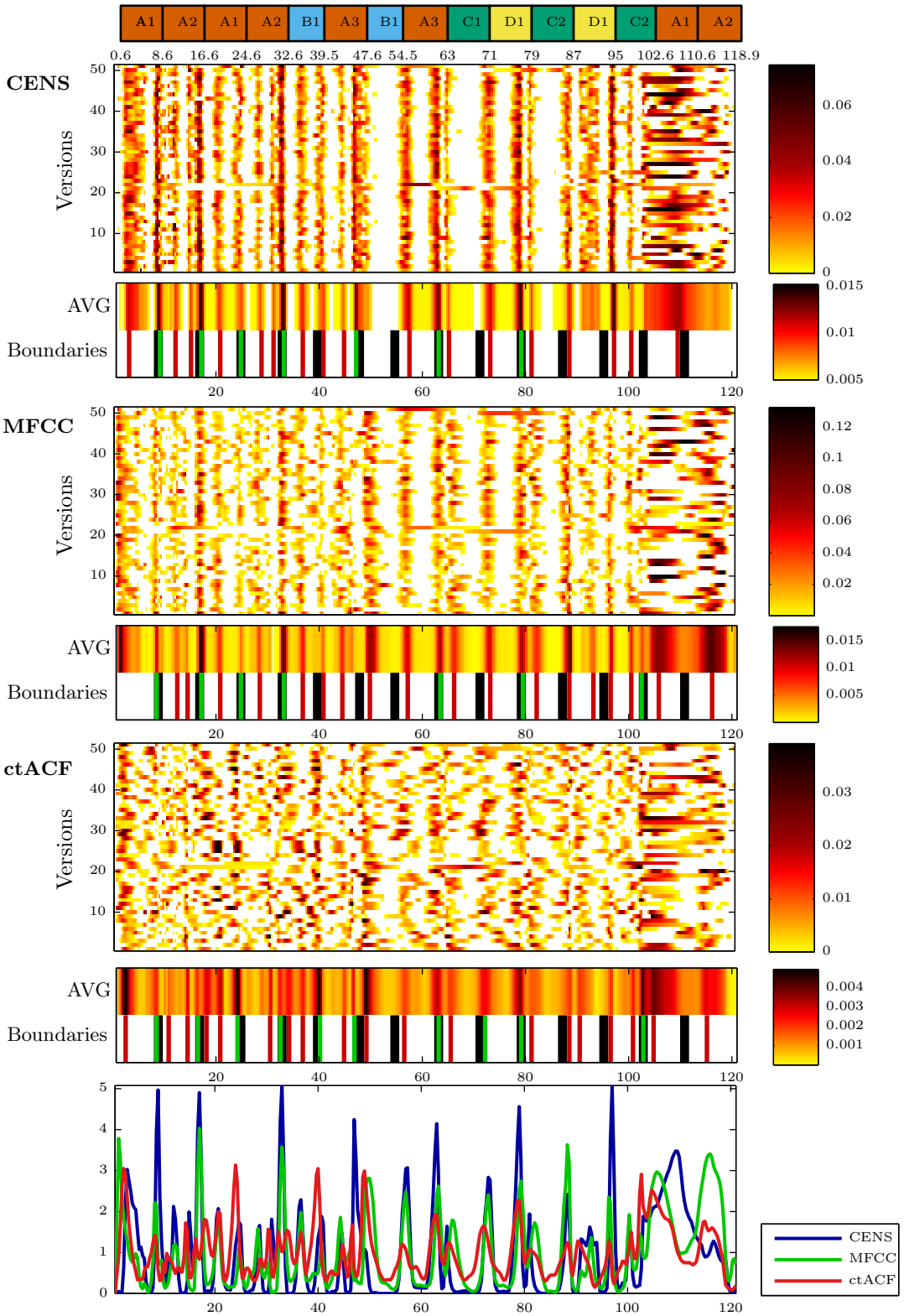
Mazurka 7 No. 1



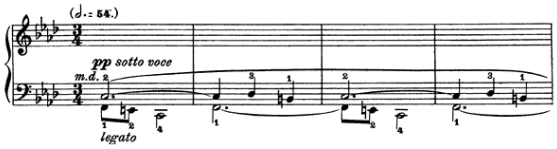


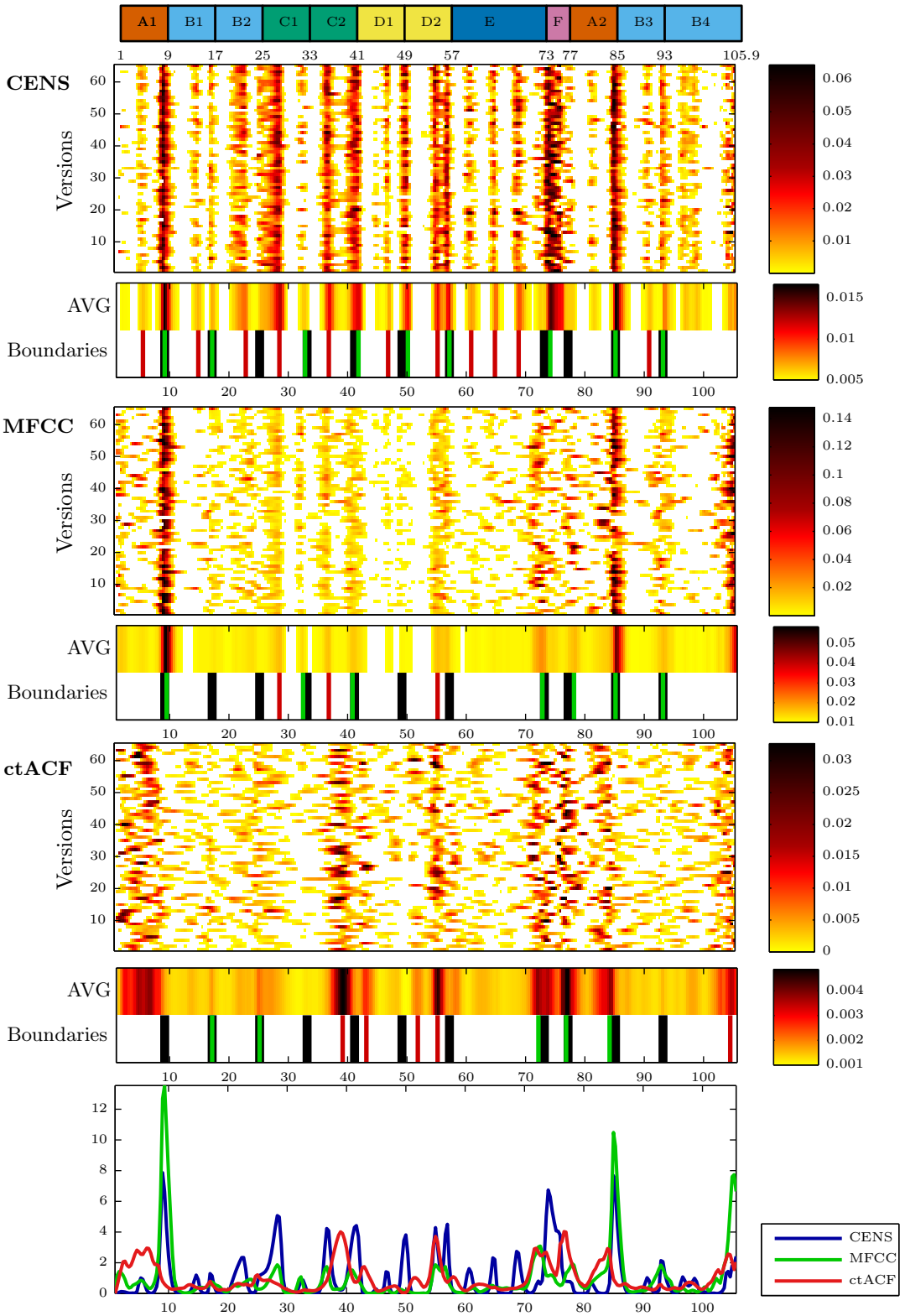
Mazurka 7 No. 2



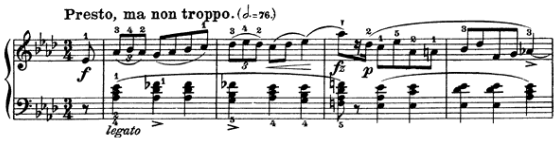


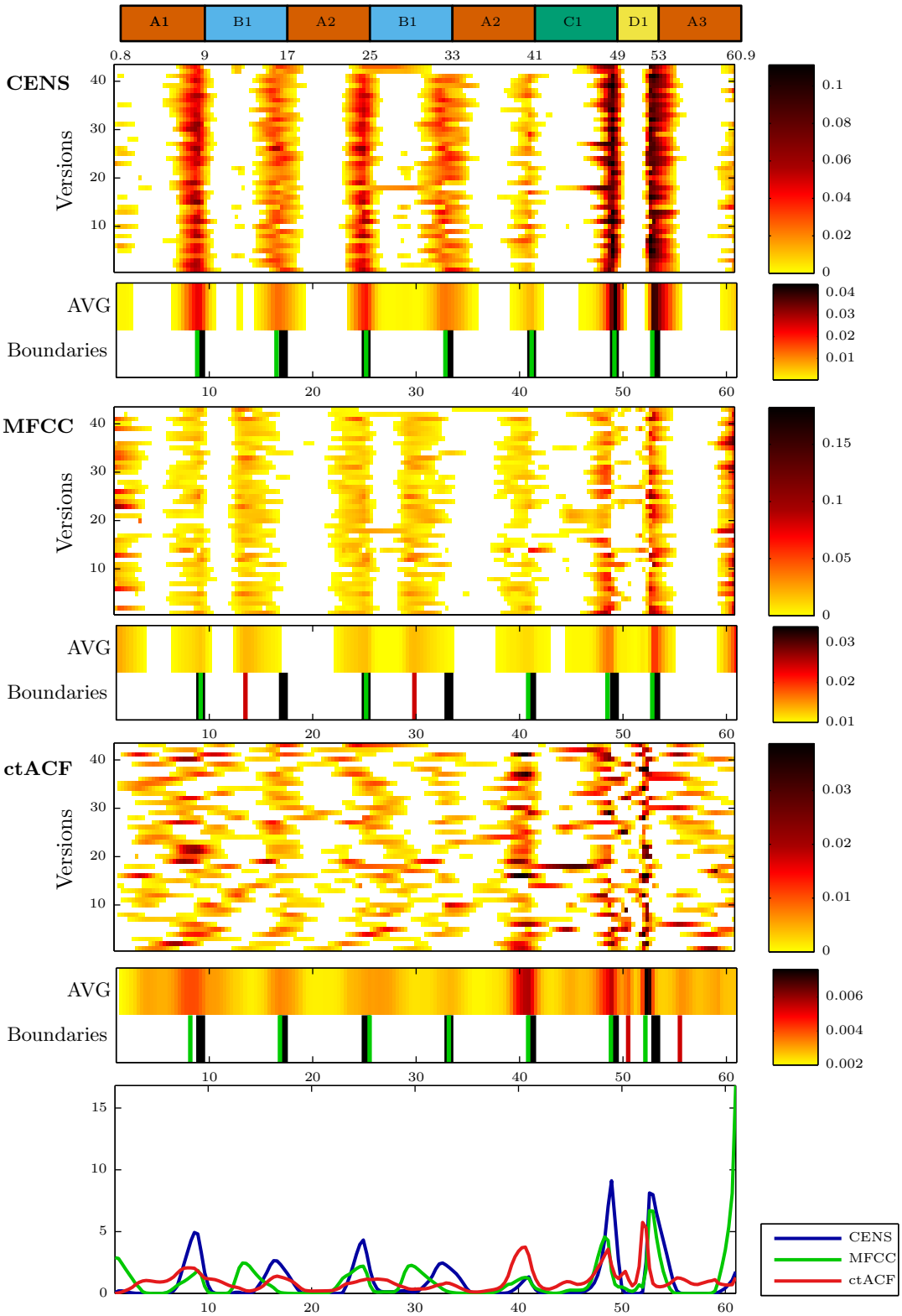
Mazurka 7 No. 3



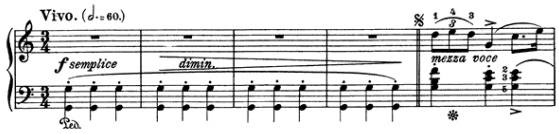


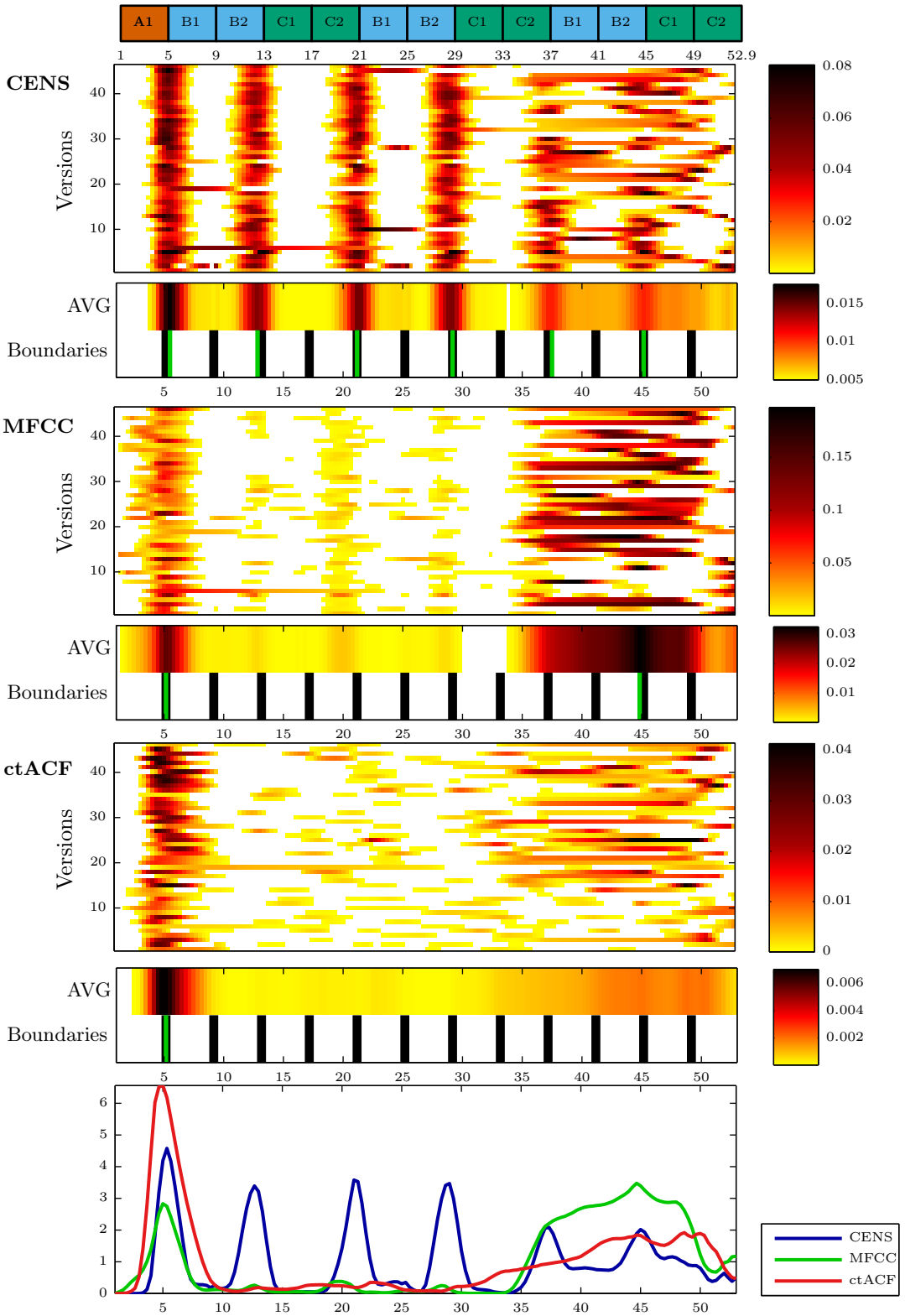
Mazurka 7 No. 4



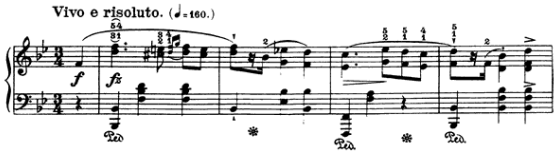


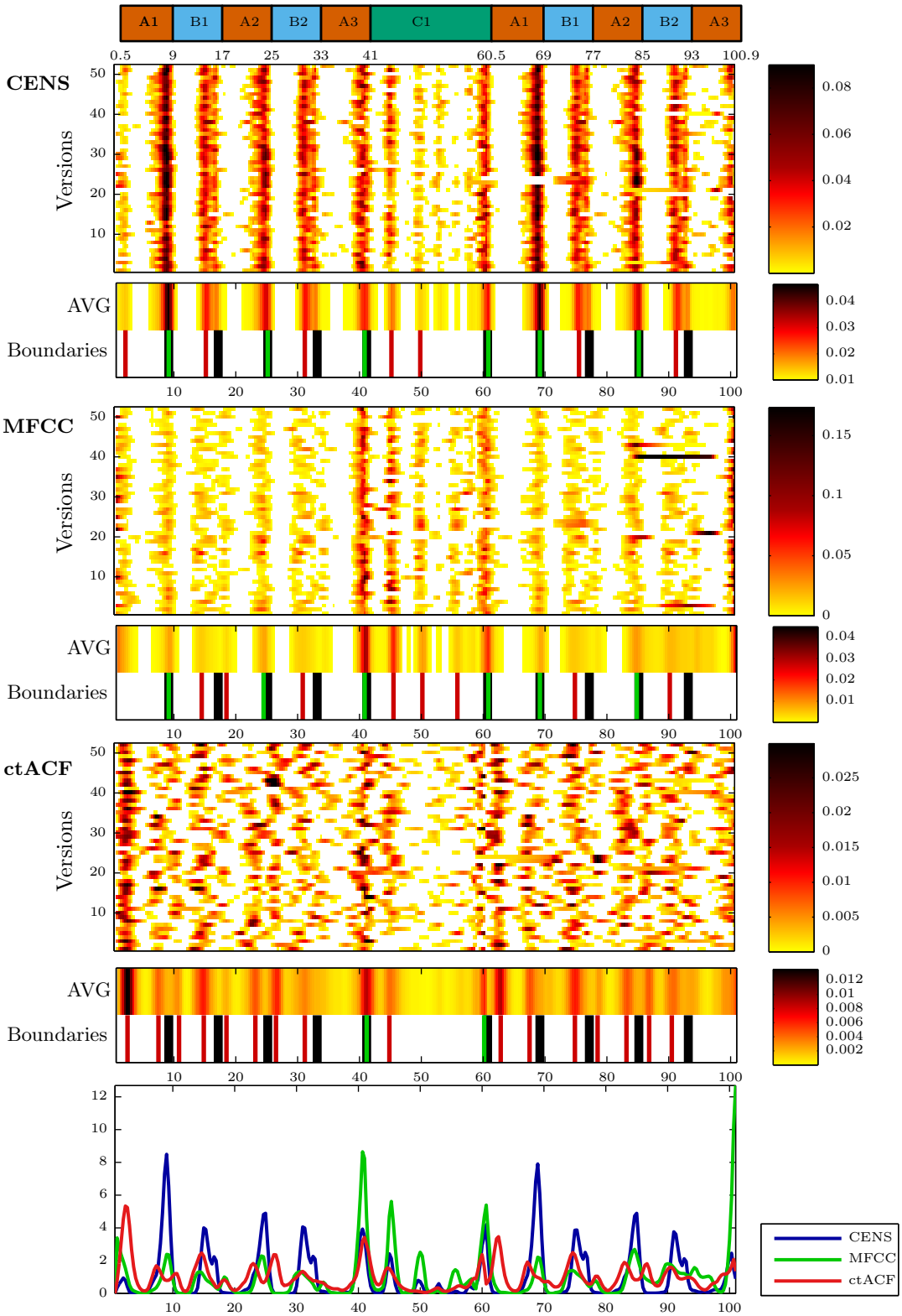
Mazurka 7 No. 5



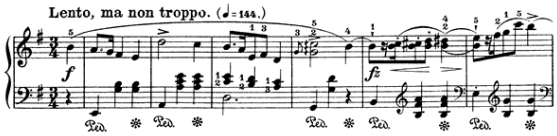


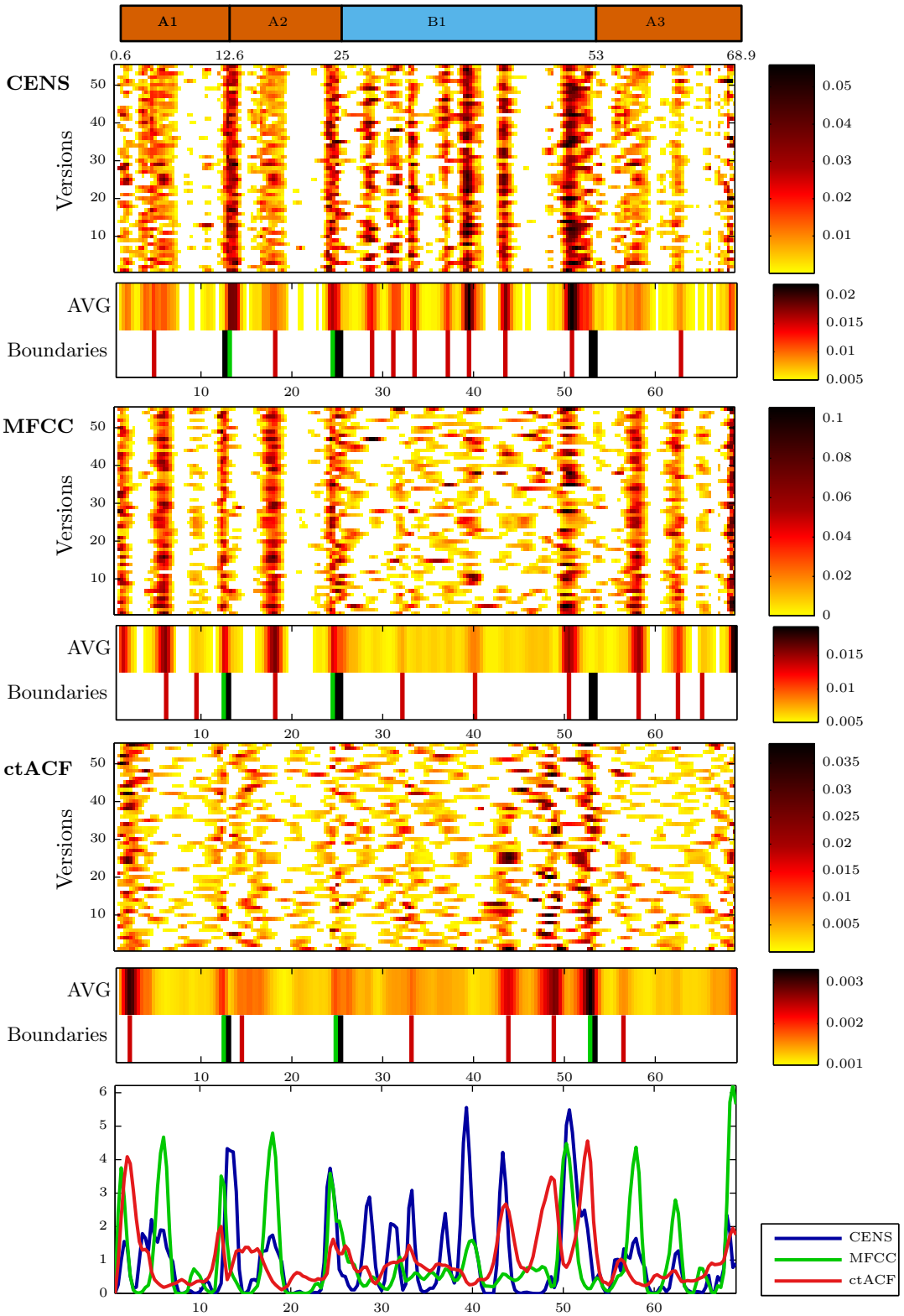
Mazurka 17 No. 1



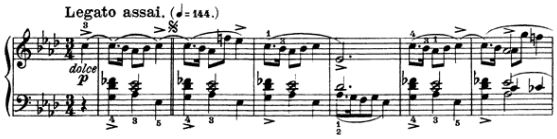


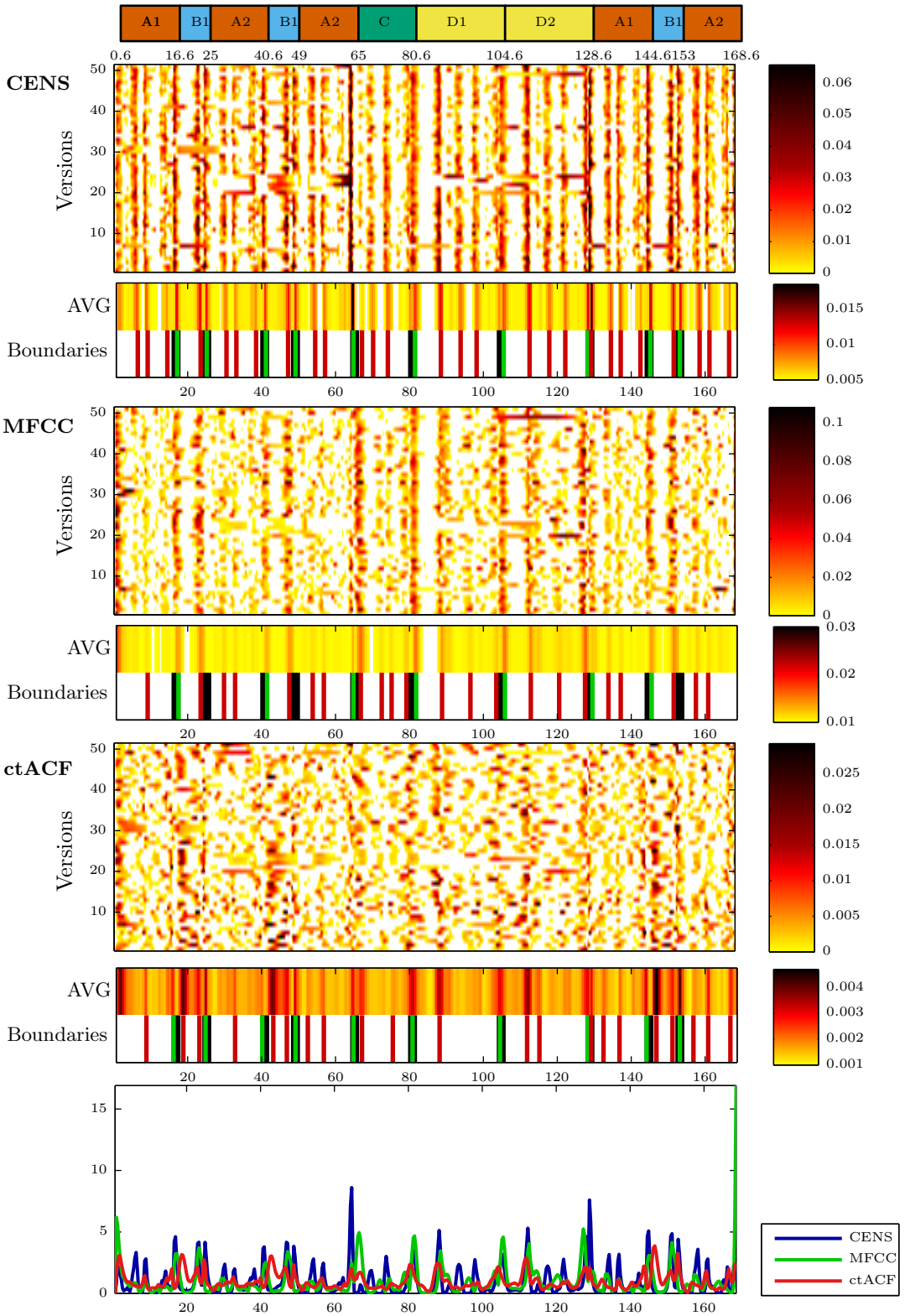
Mazurka 17 No. 2



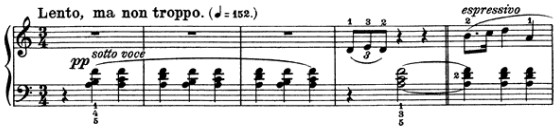


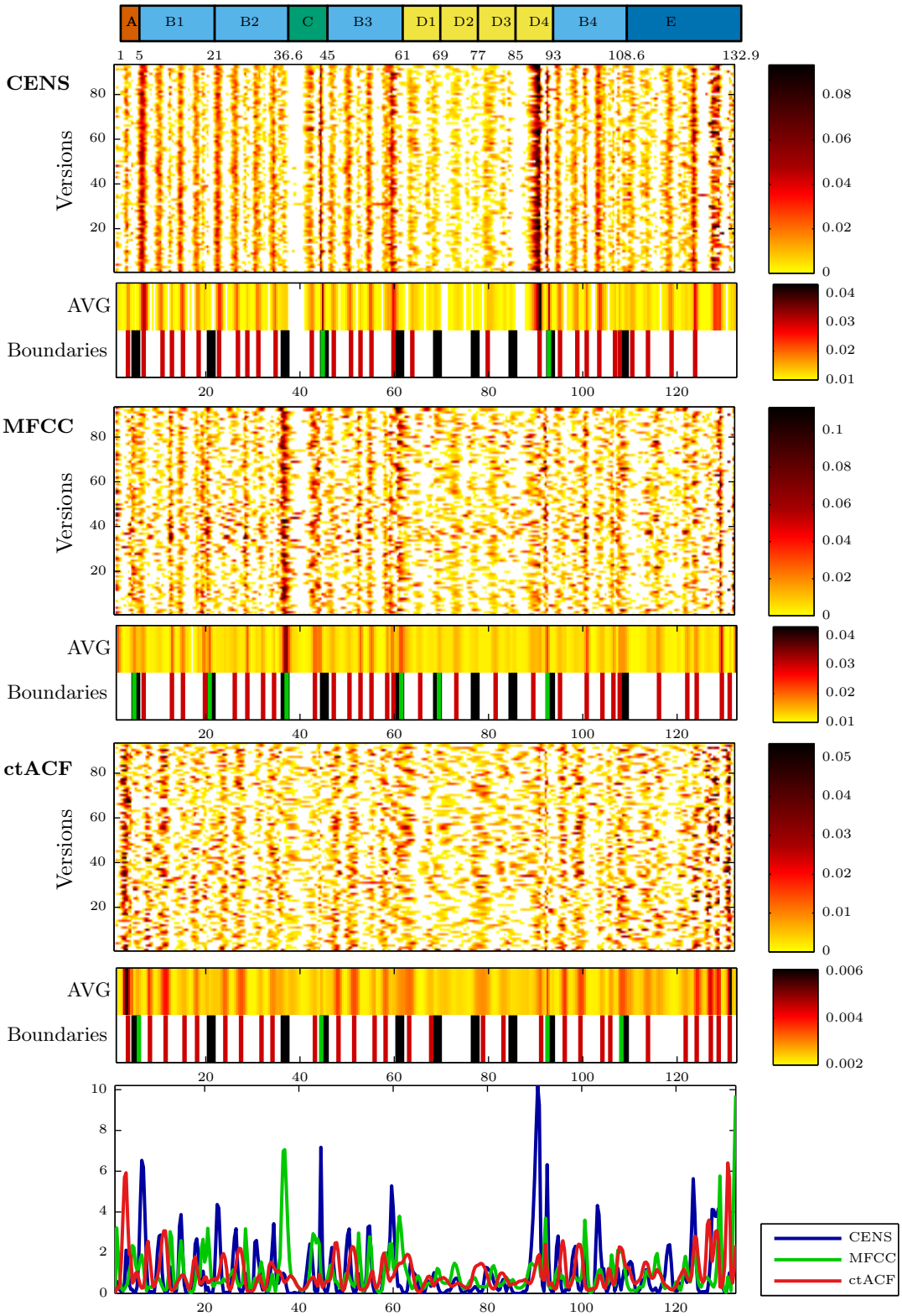
Mazurka 17 No. 3



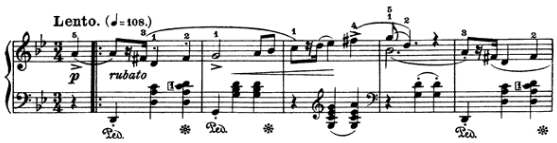


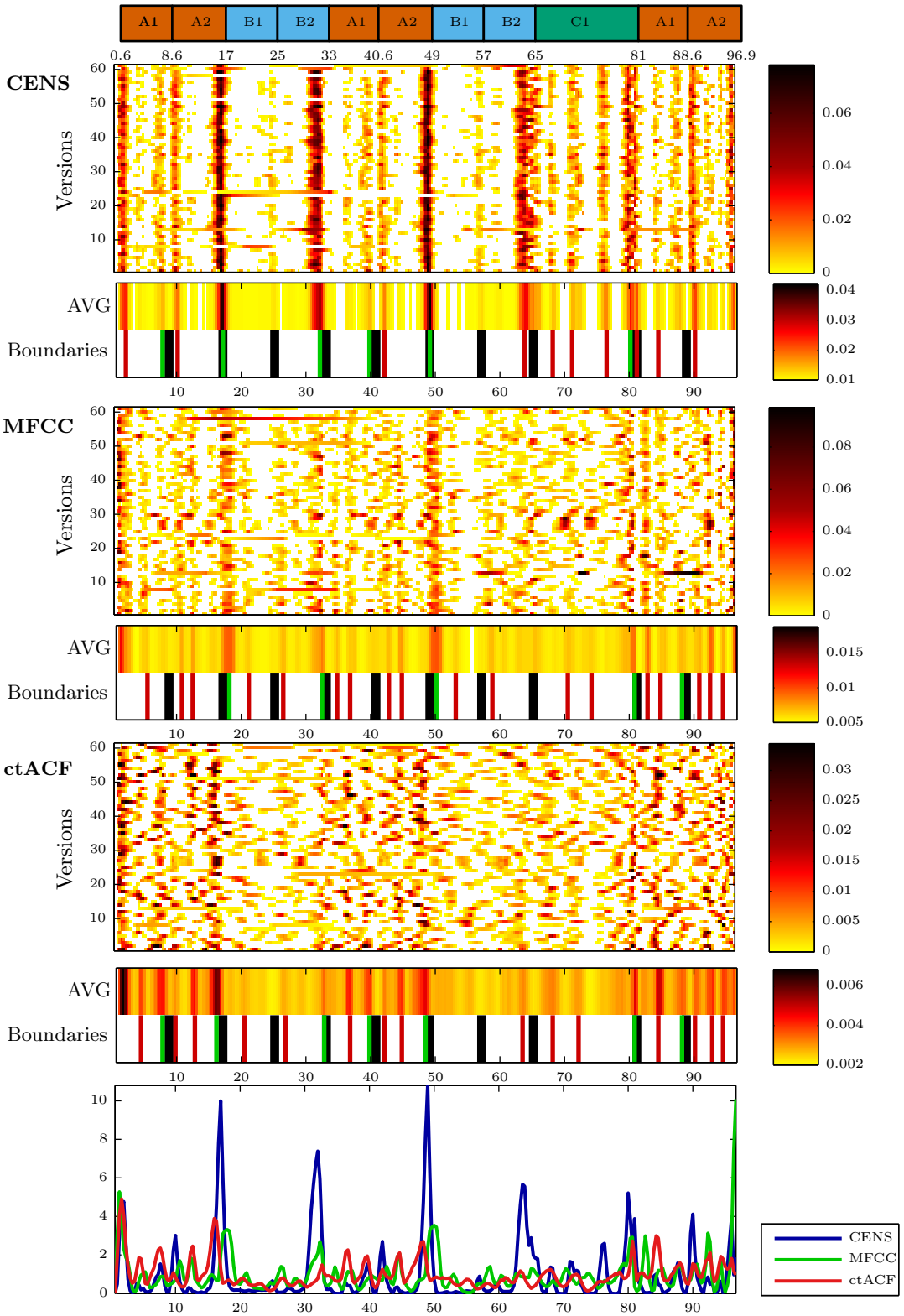
Mazurka 17 No. 4



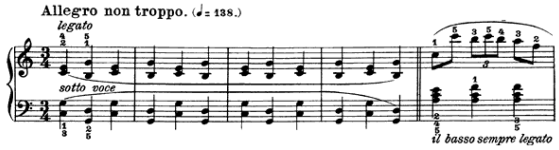


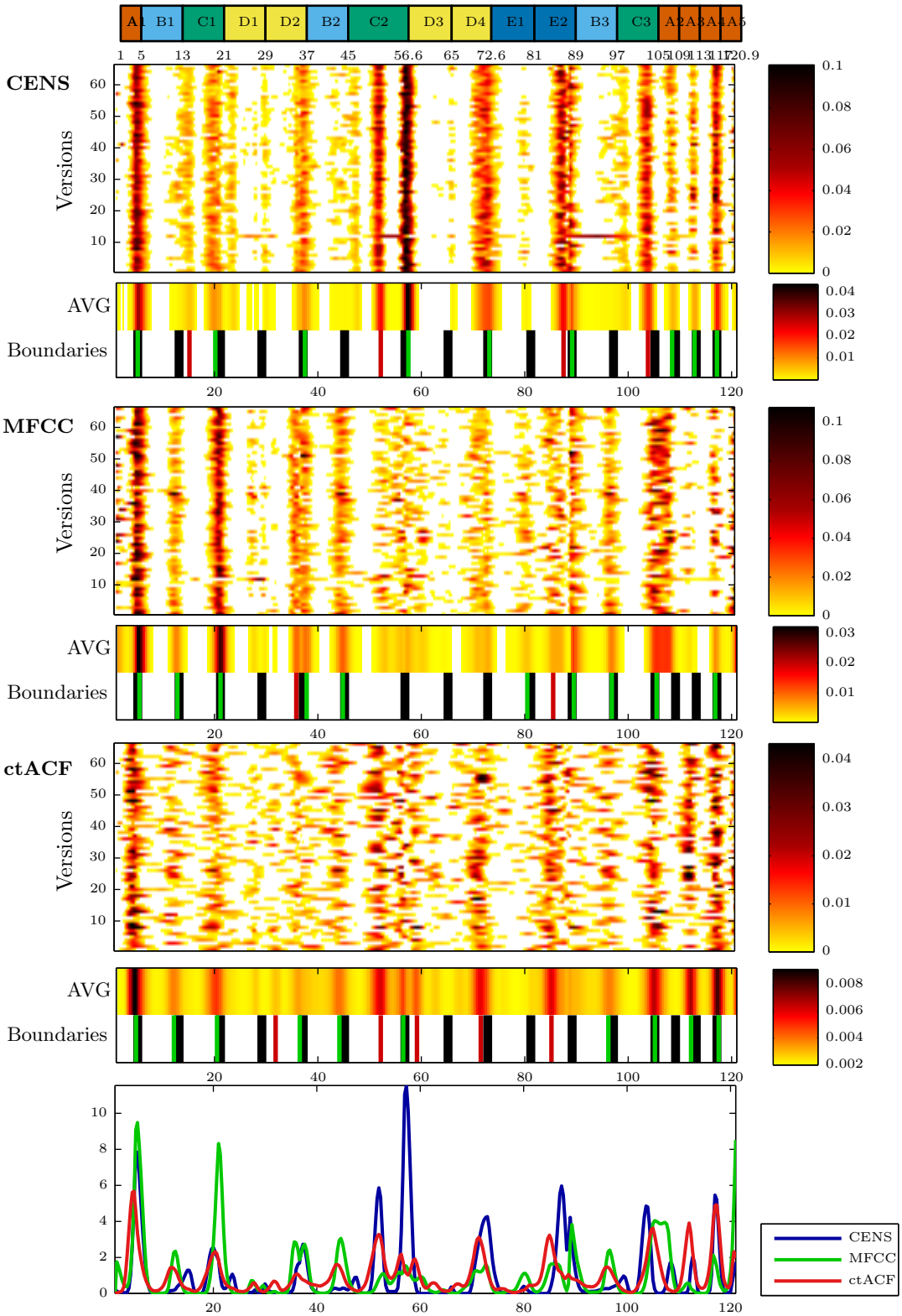
Mazurka 24 No. 1



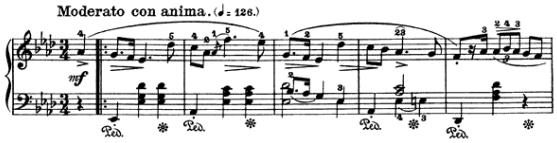


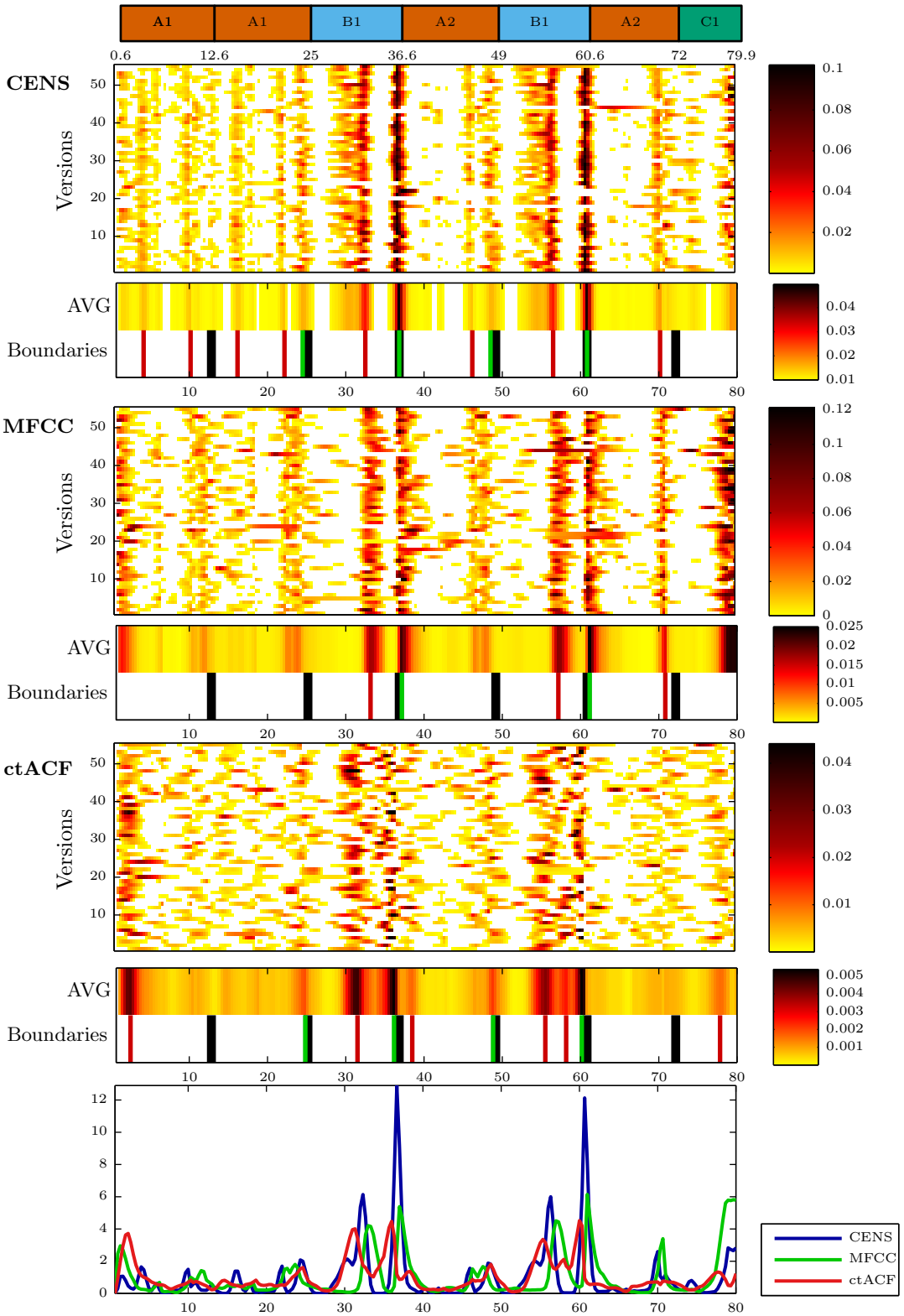
Mazurka 24 No. 2



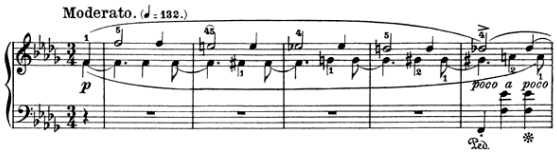


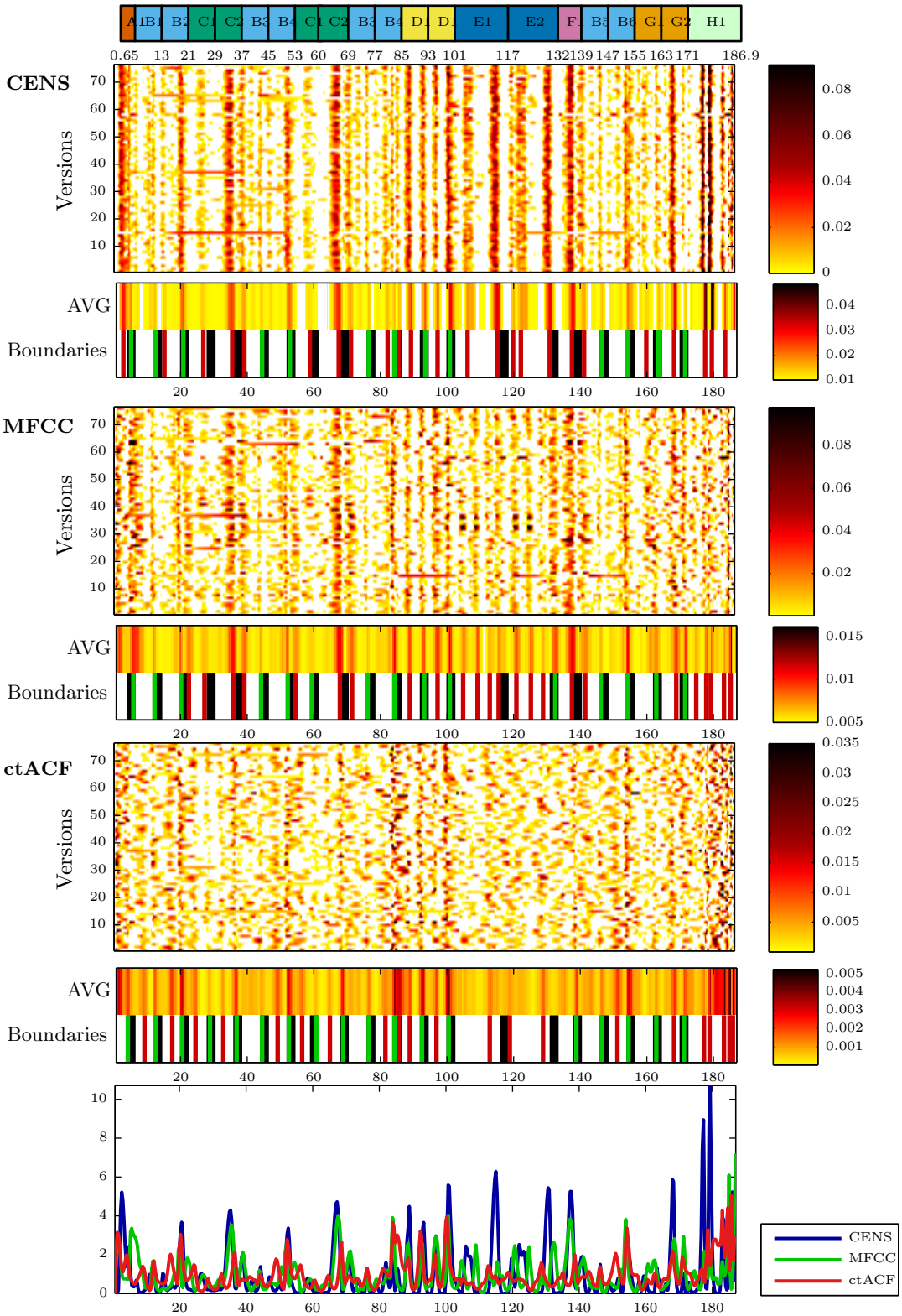
Mazurka 24 No. 3



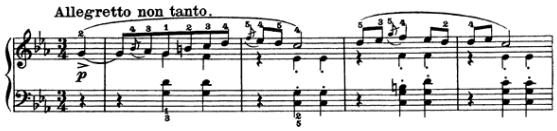


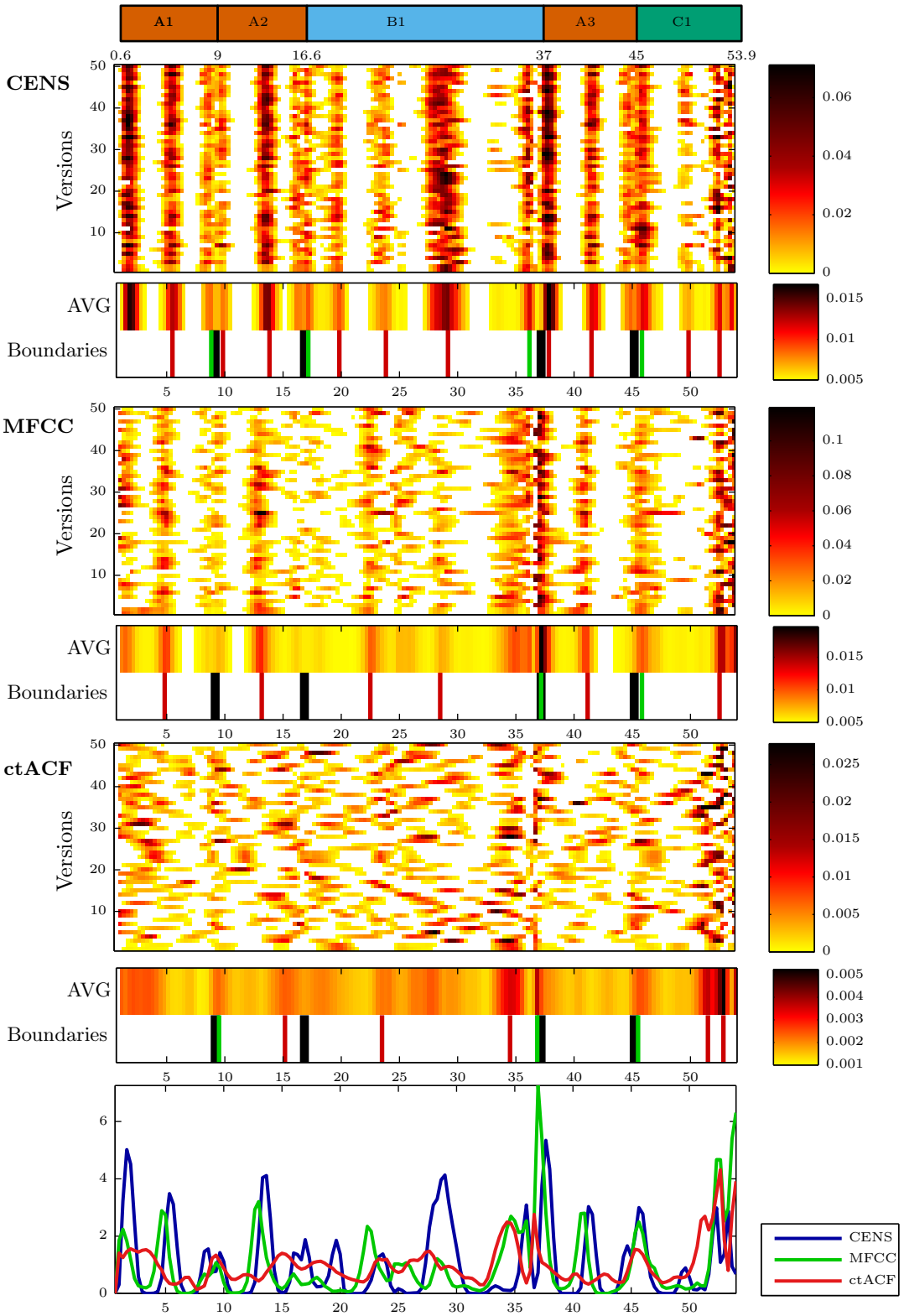
Mazurka 24 No. 4



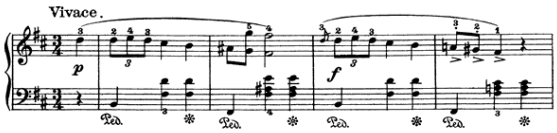


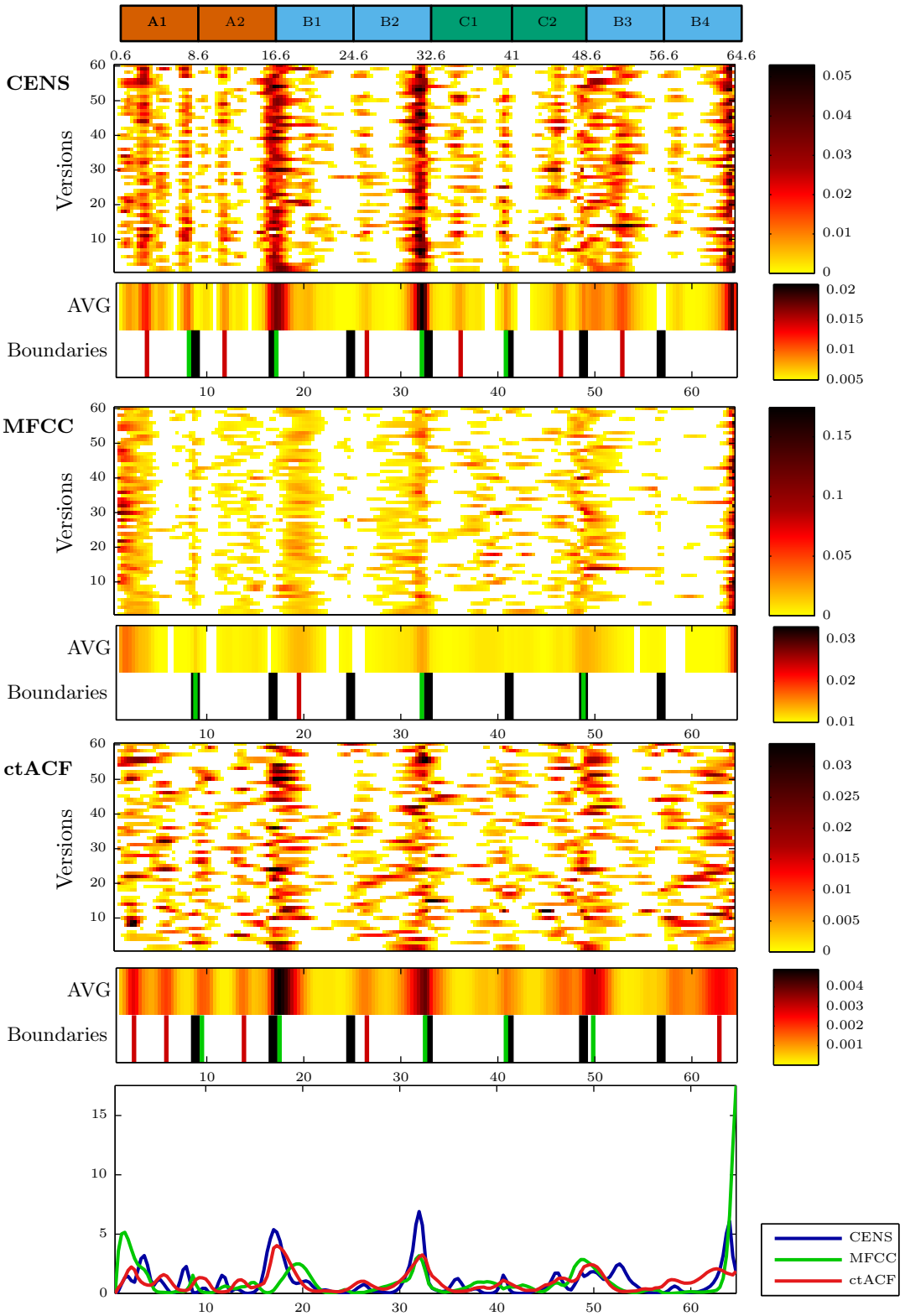
Mazurka 30 No. 1



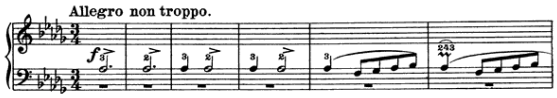


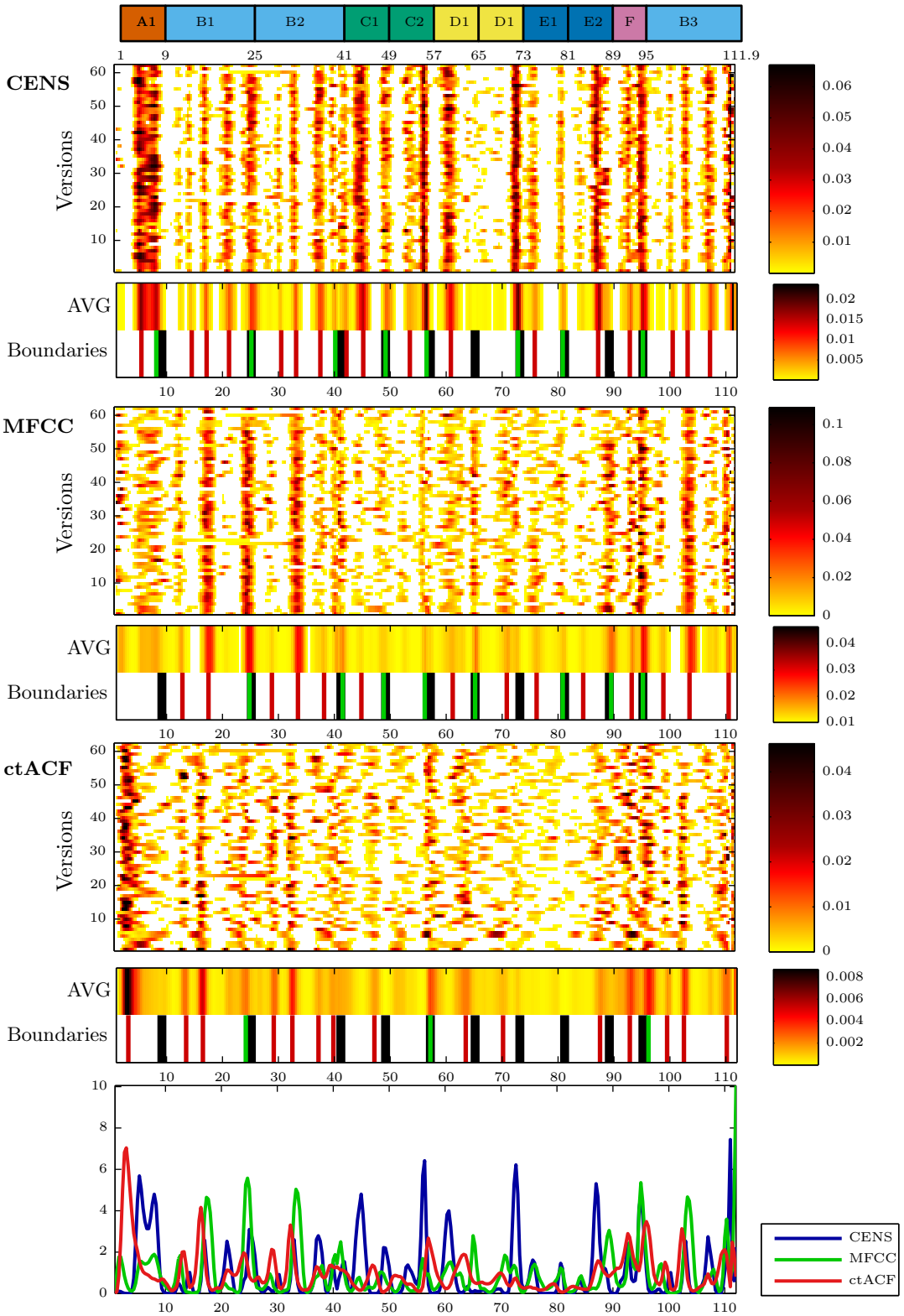
Mazurka 30 No. 2



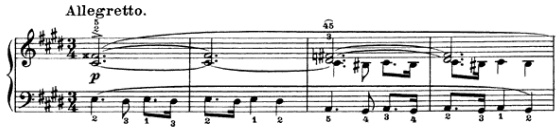


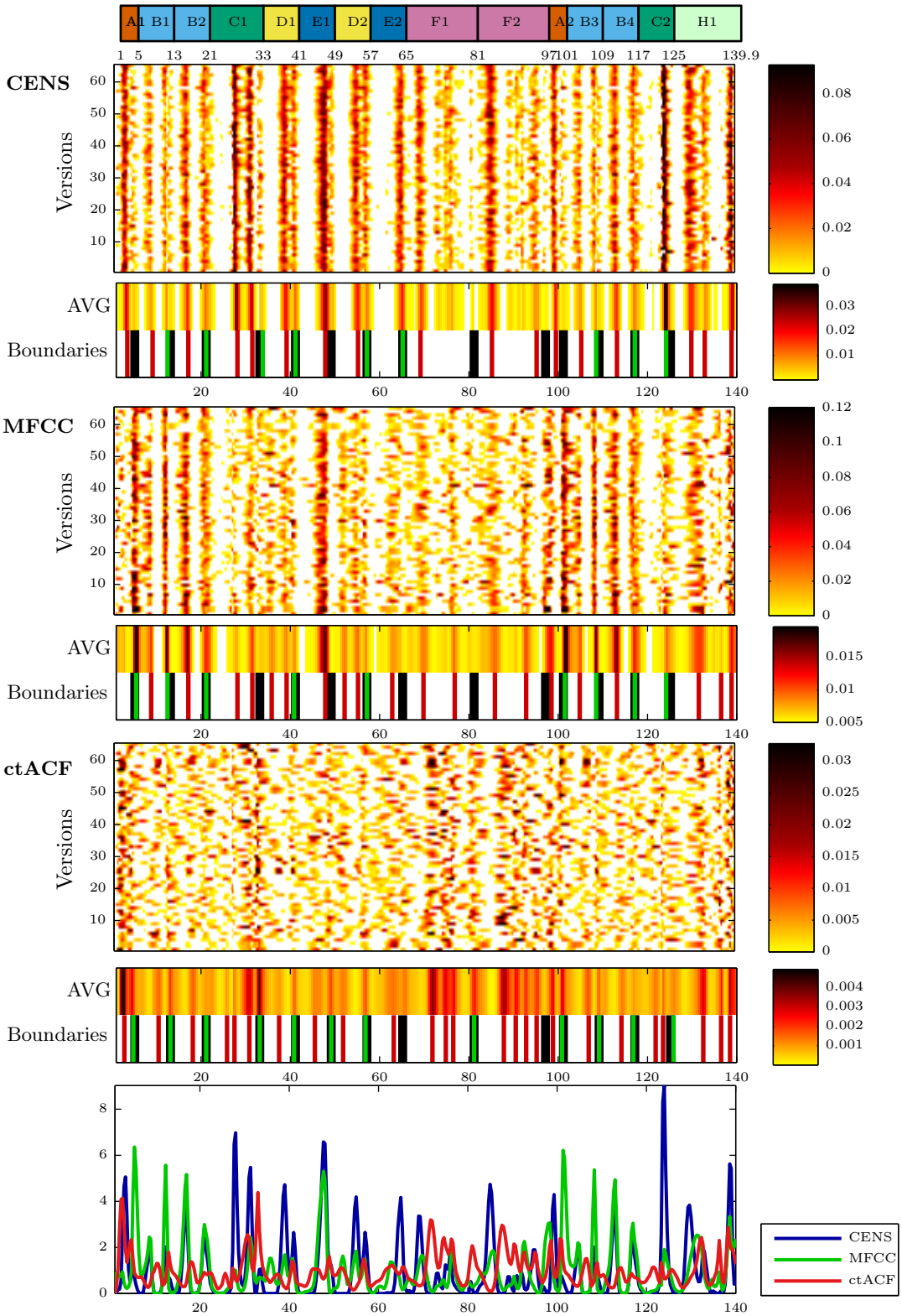
Mazurka 30 No. 3



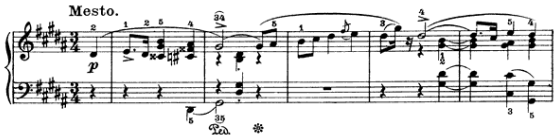


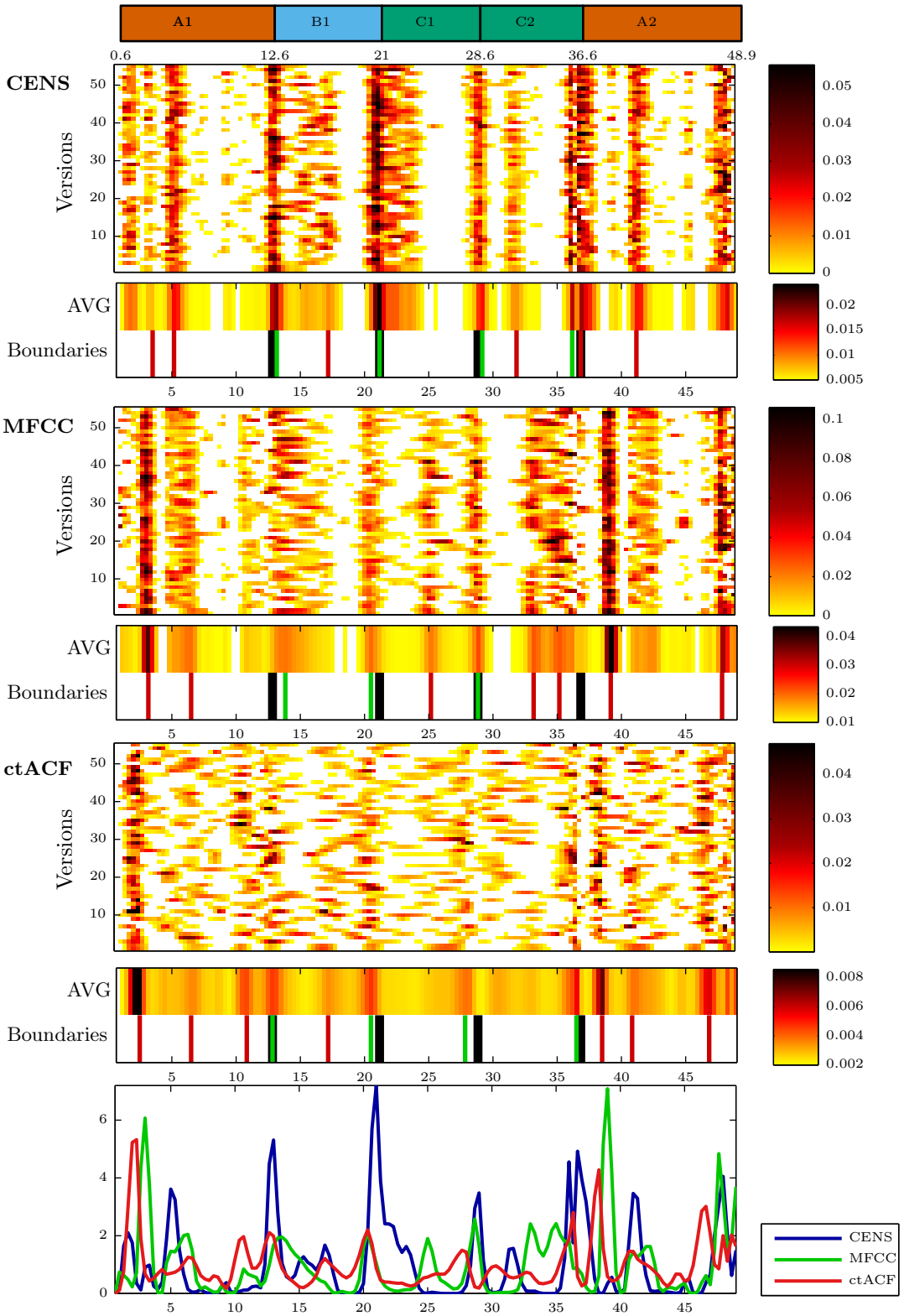
Mazurka 30 No. 4



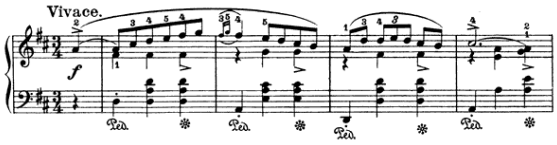


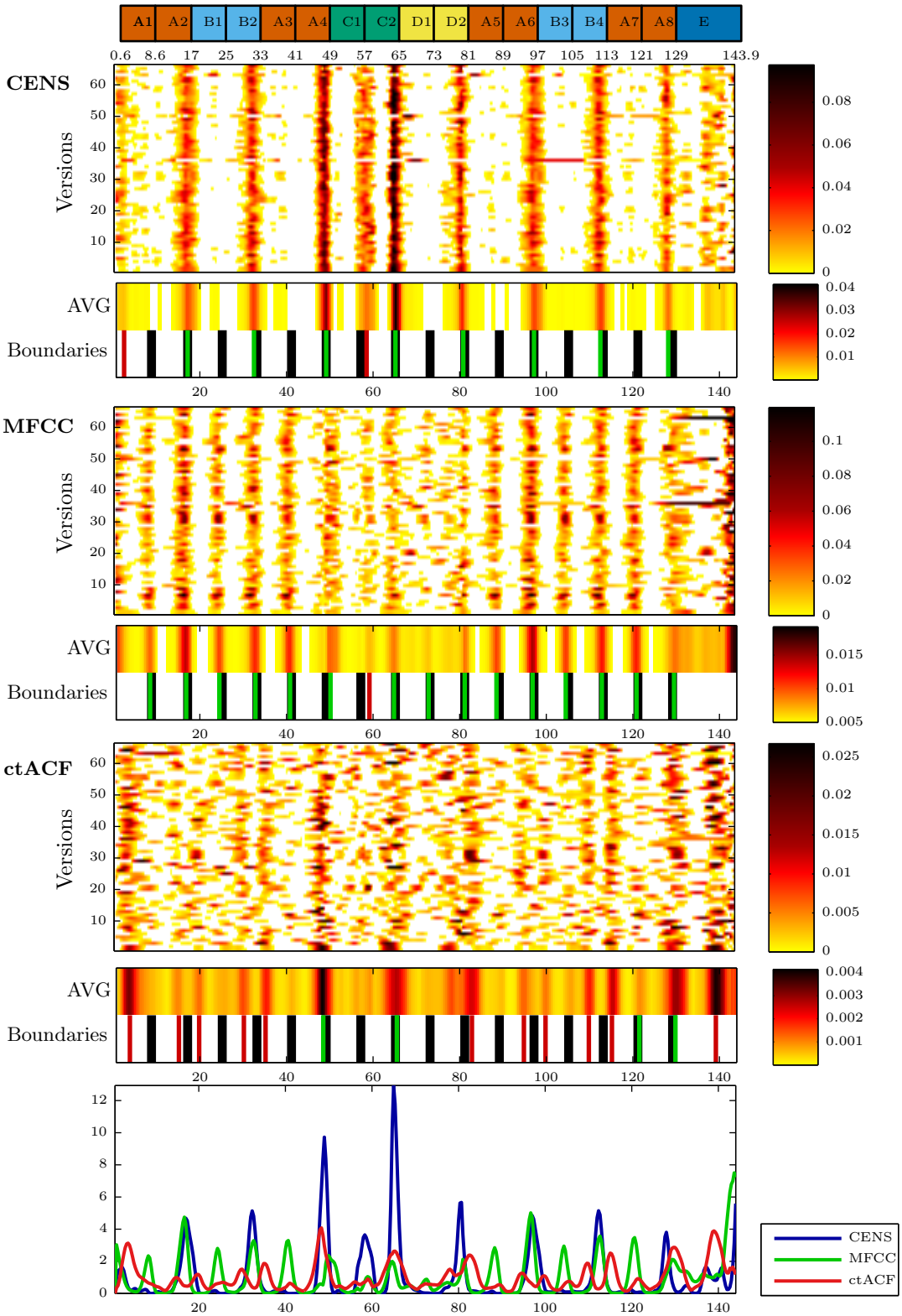
Mazurka 33 No. 1



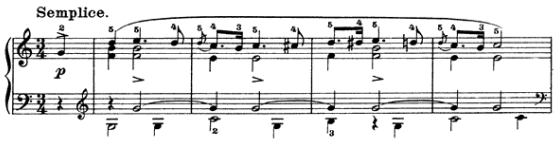


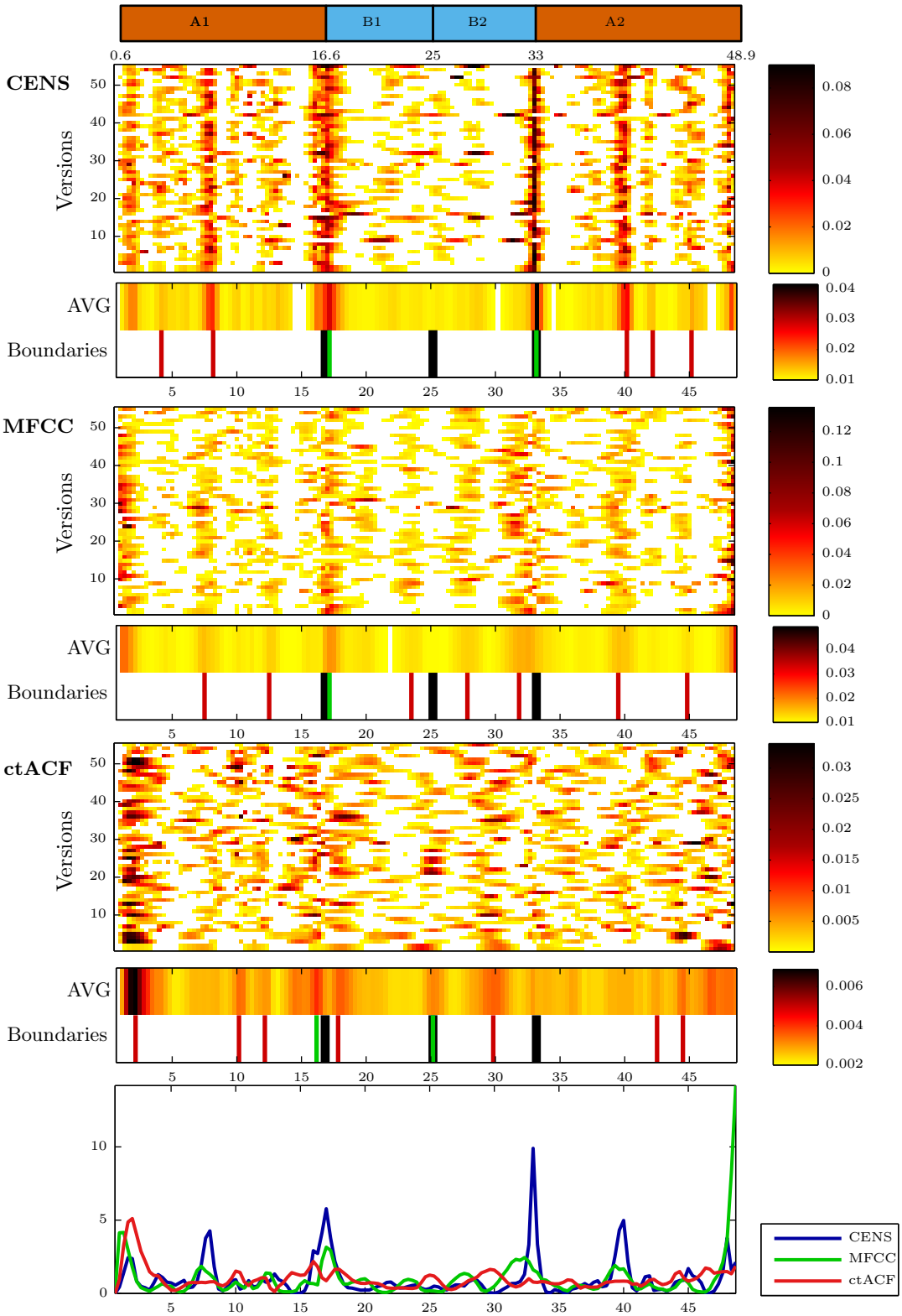
Mazurka 33 No. 2



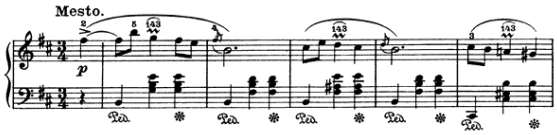


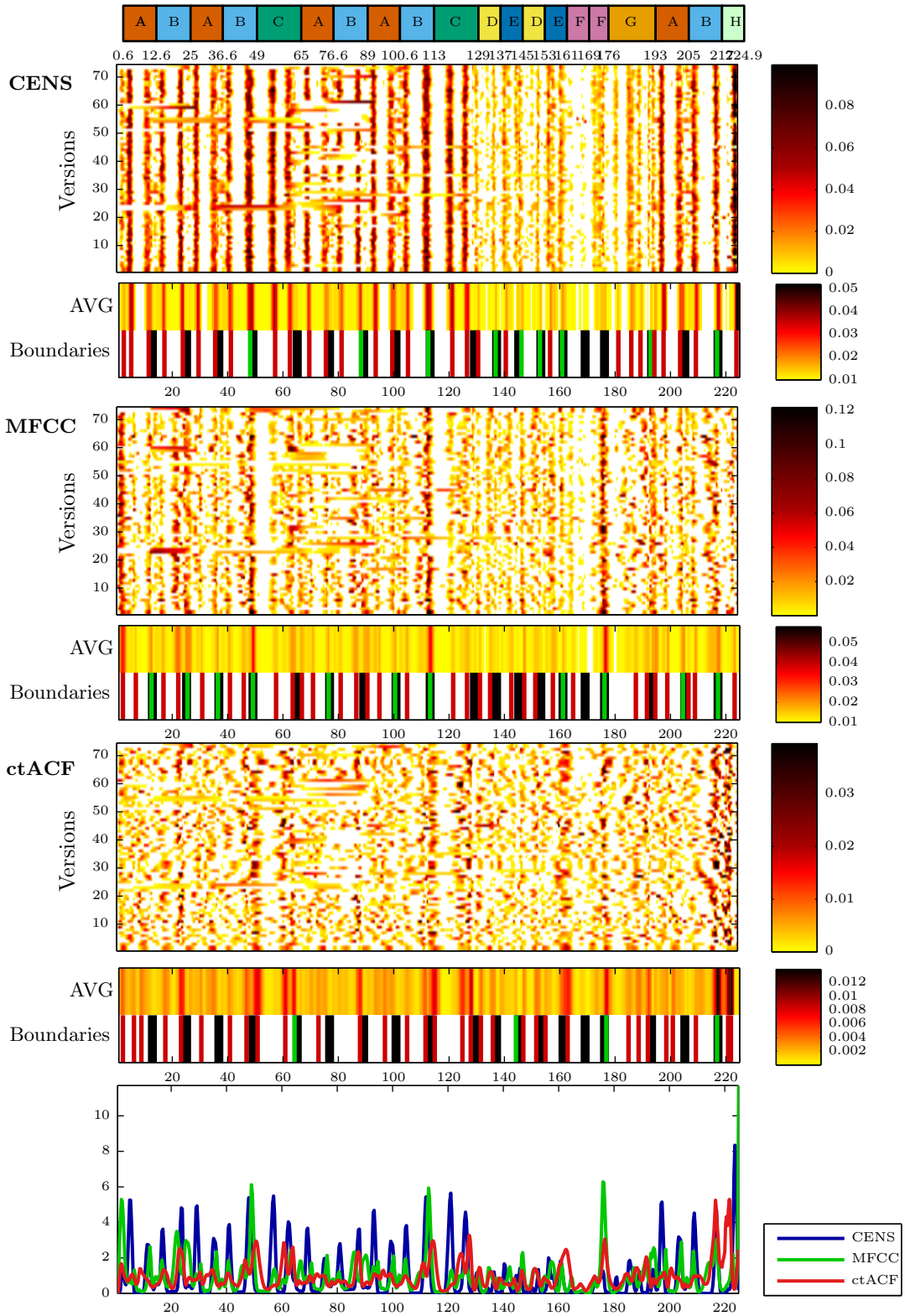
Mazurka 33 No. 3



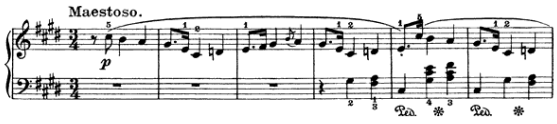


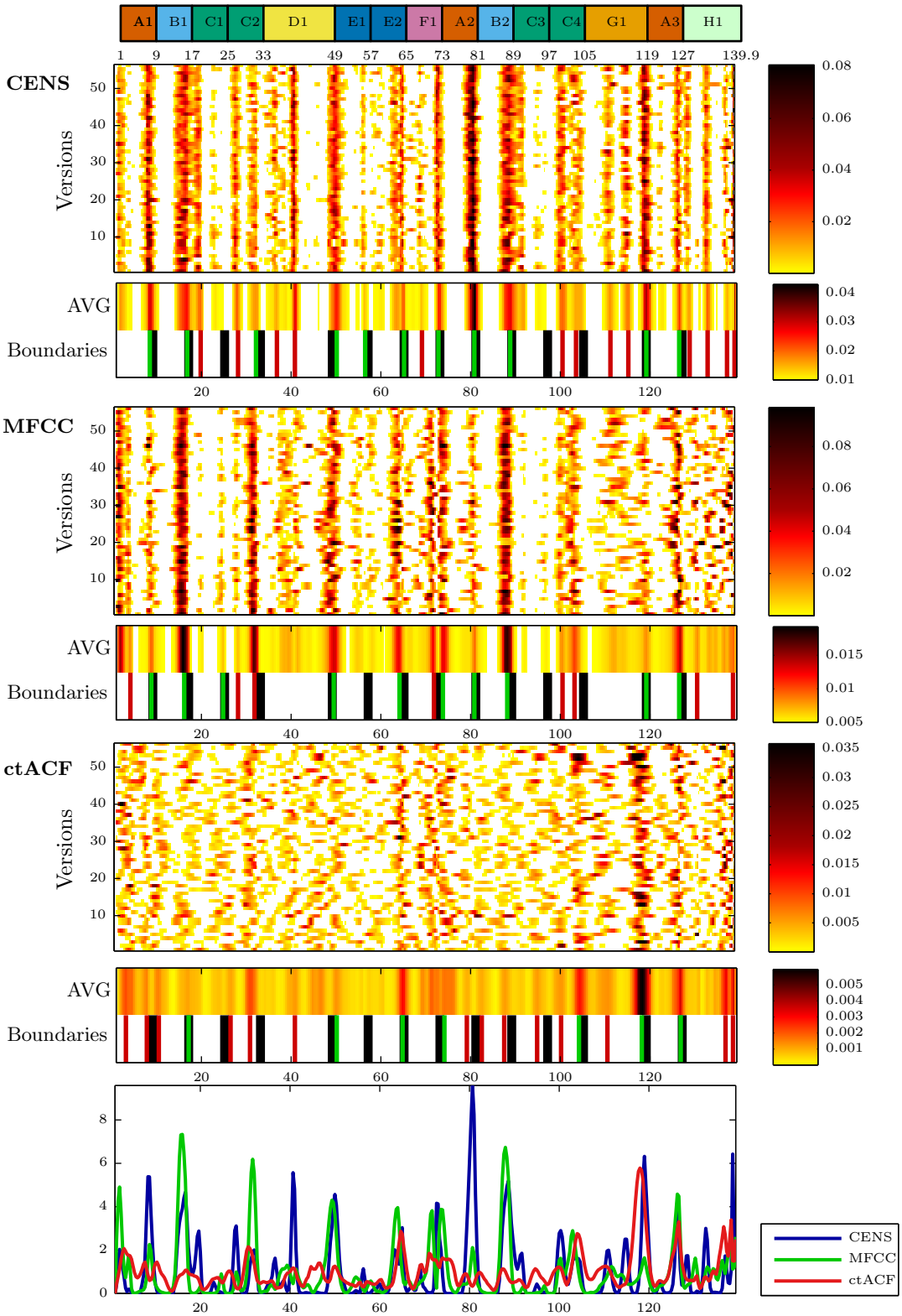
Mazurka 33 No. 4



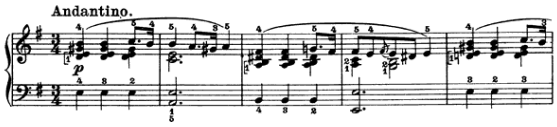


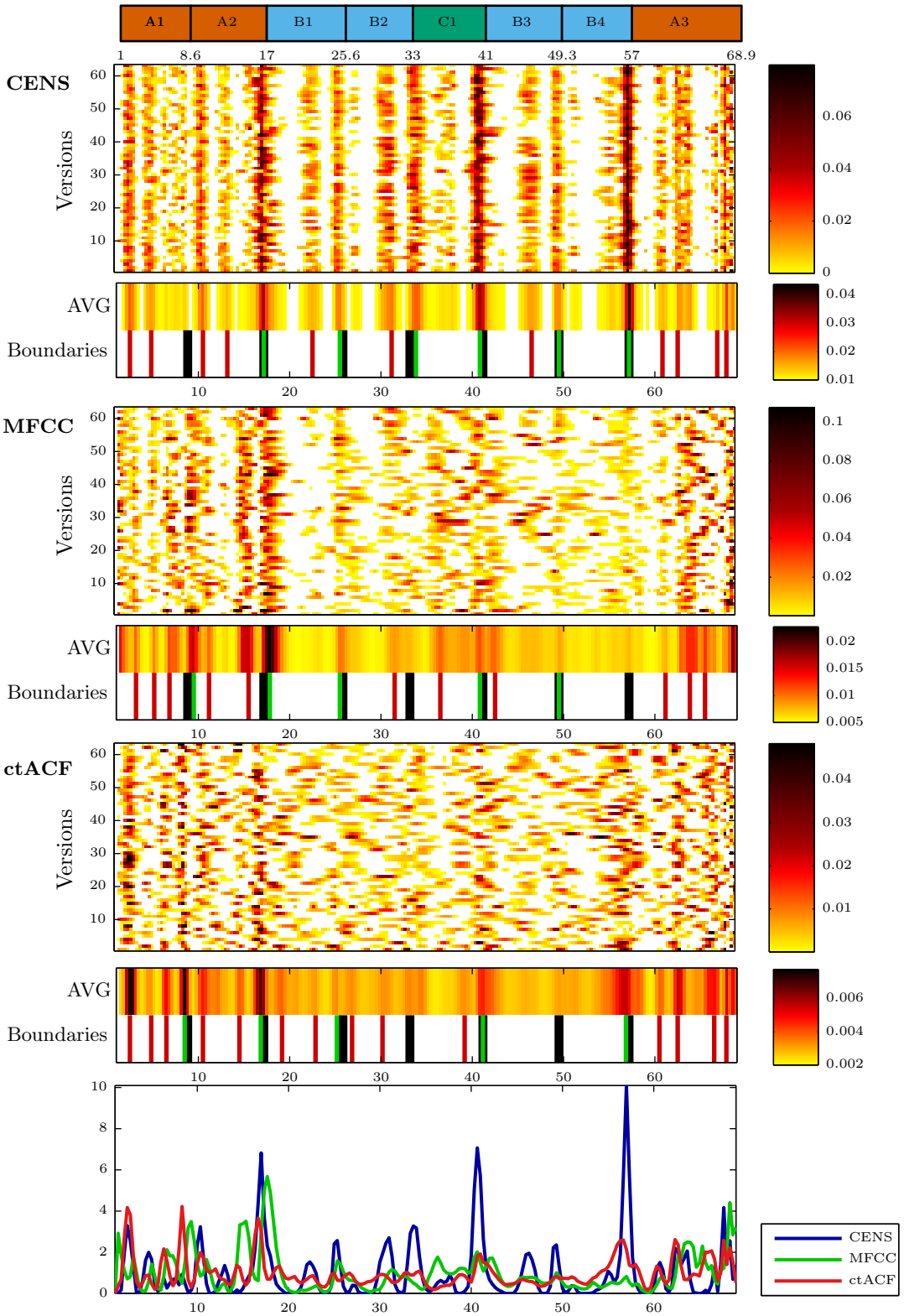
Mazurka 41 No. 1



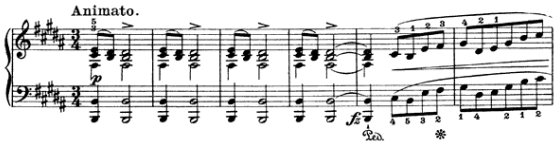


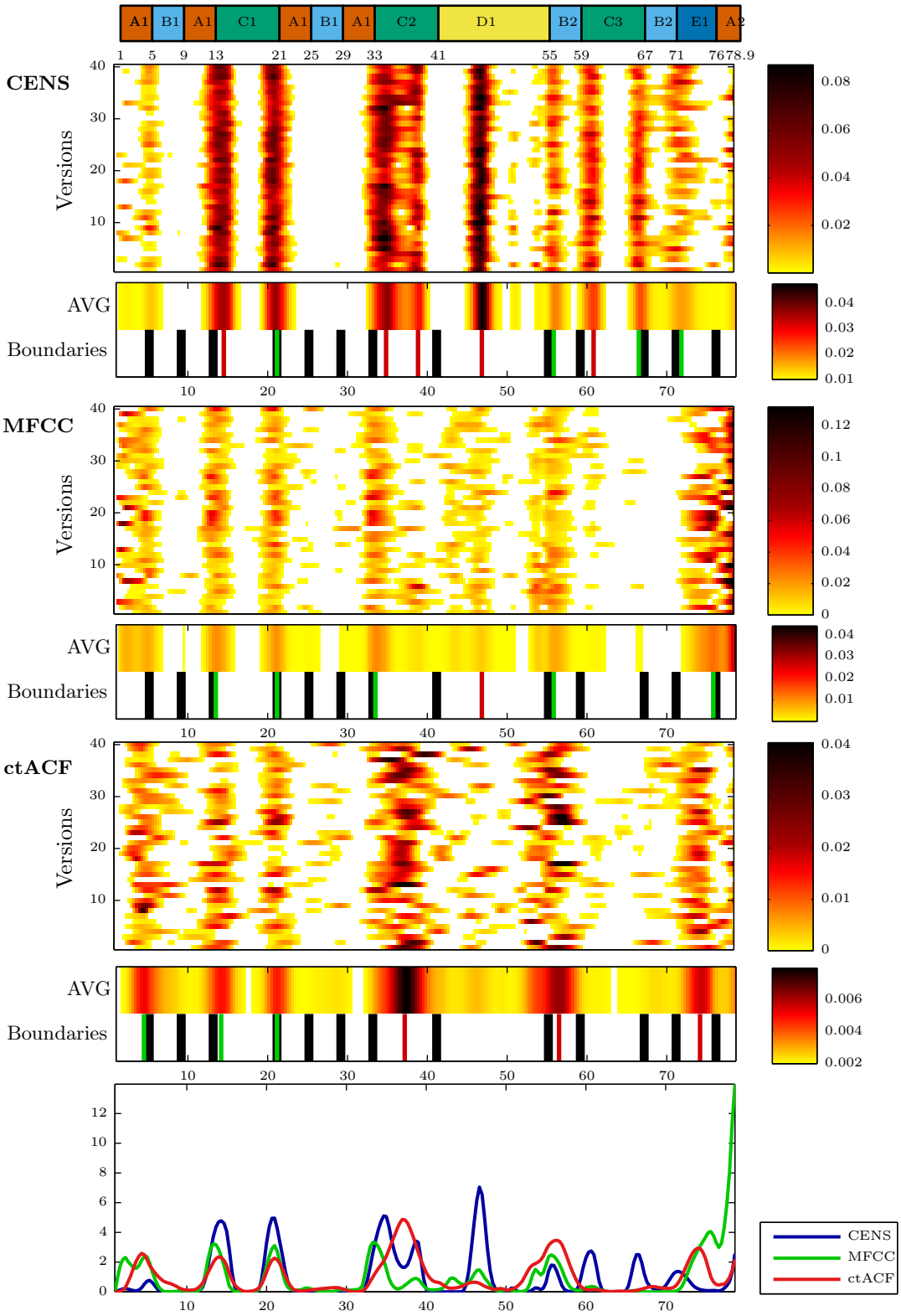
Mazurka 41 No. 2



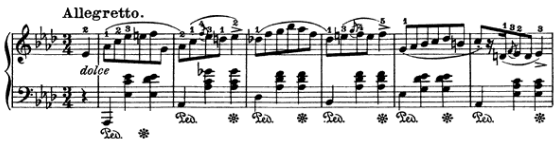


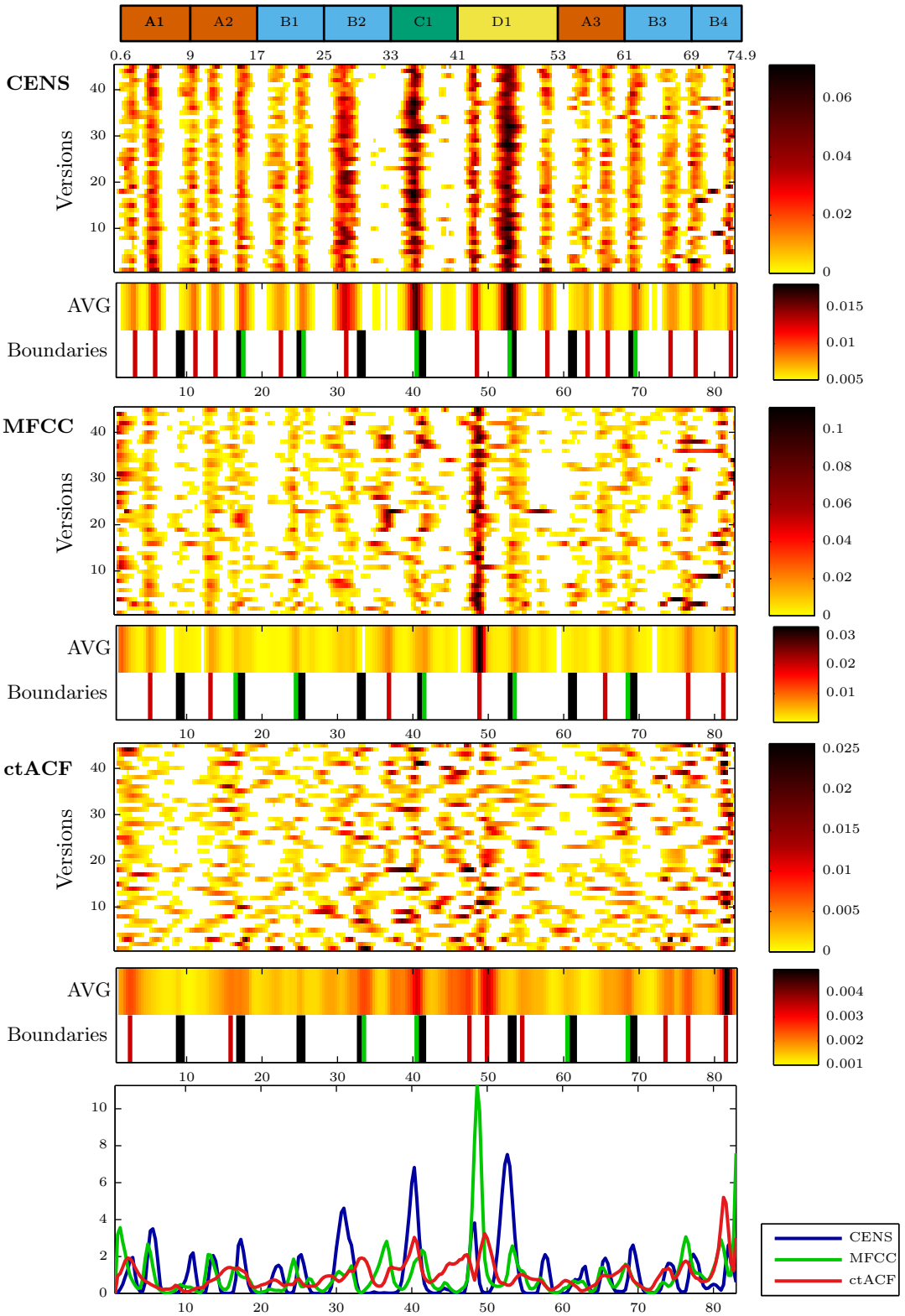
Mazurka 41 No. 3



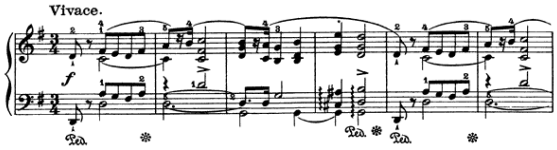


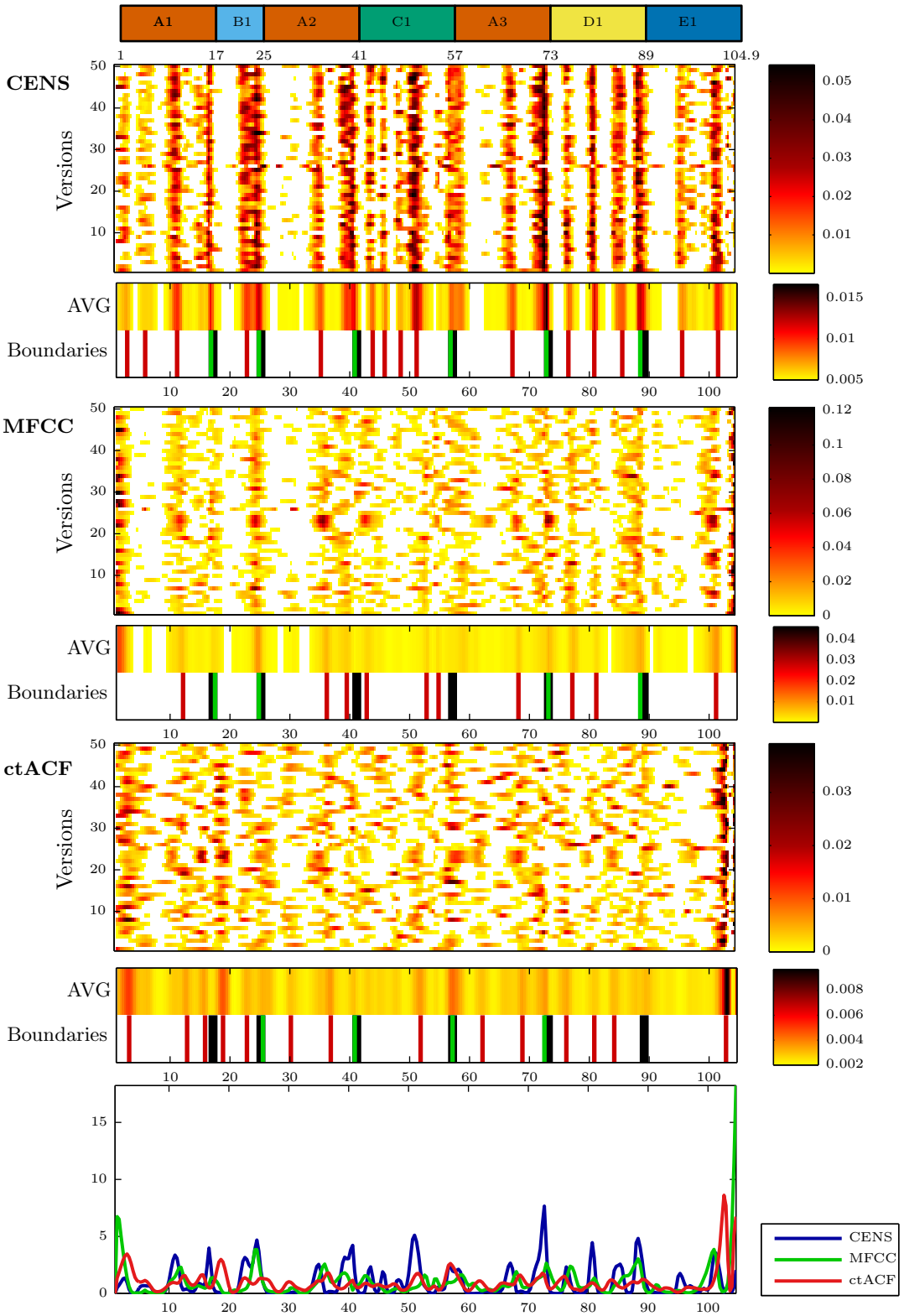
Mazurka 41 No. 4



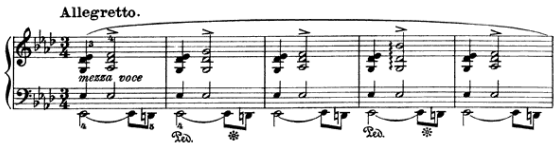


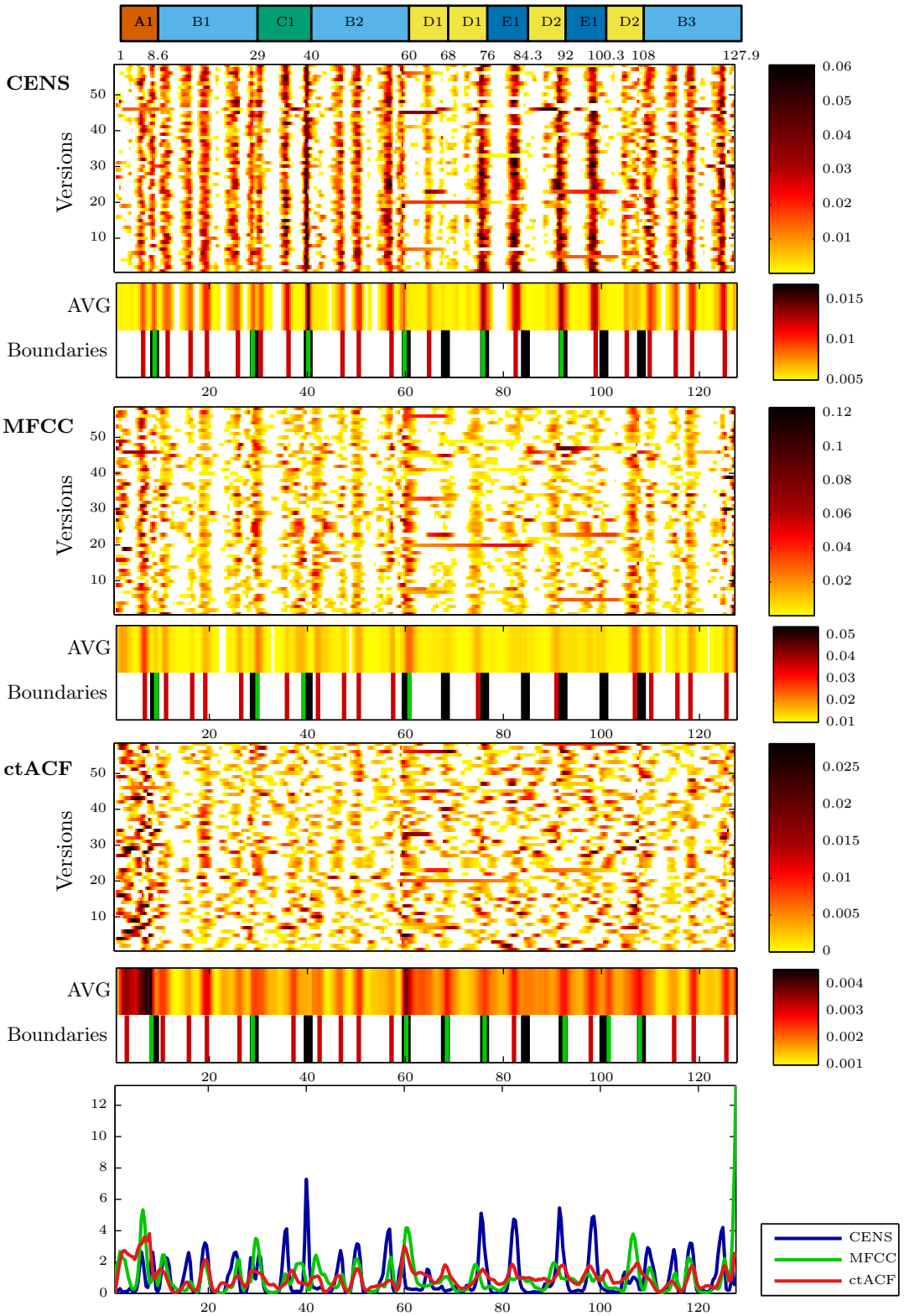
Mazurka 50 No. 1



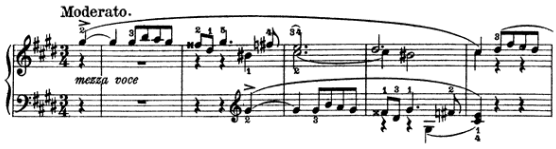


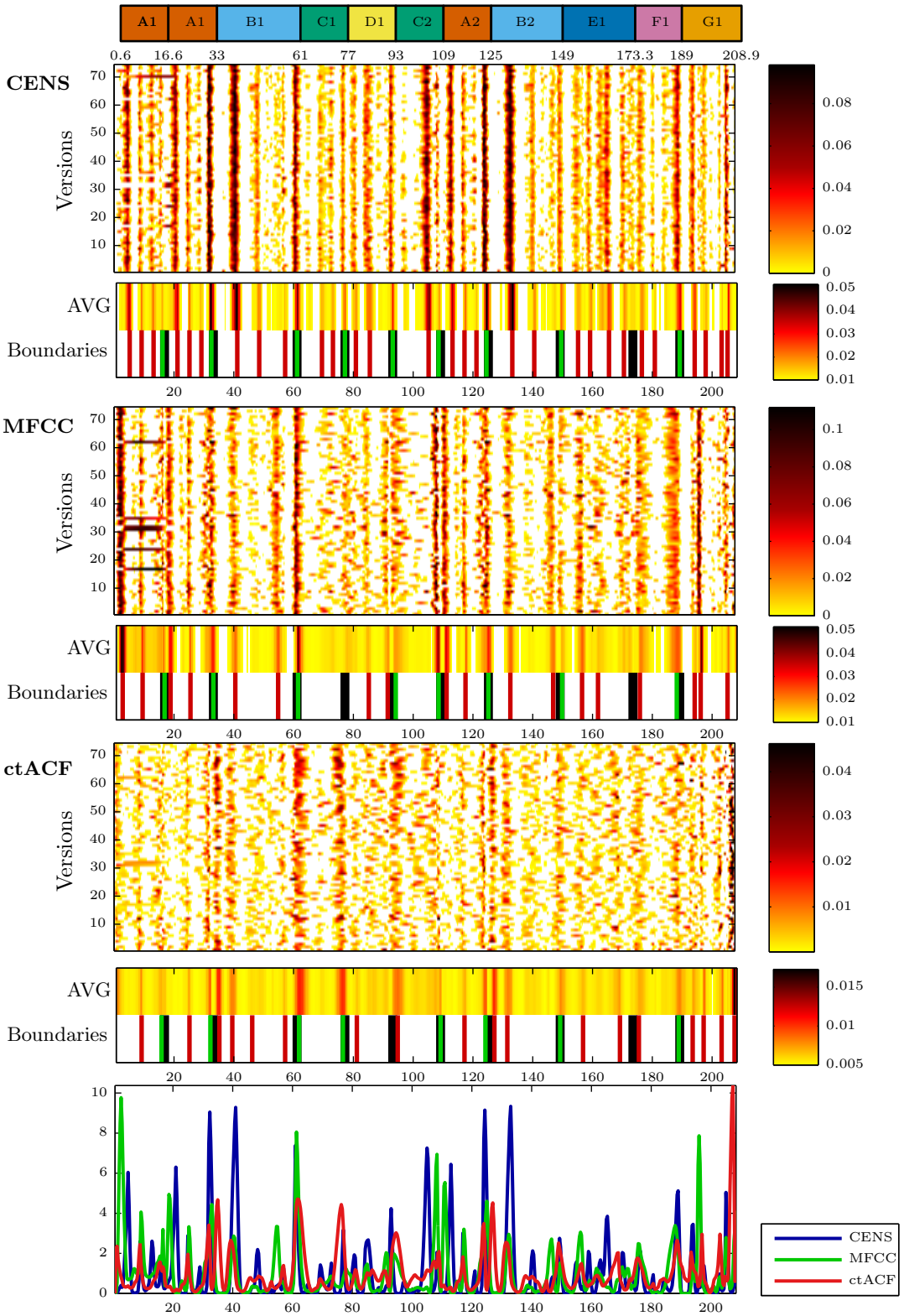
Mazurka 50 No. 2



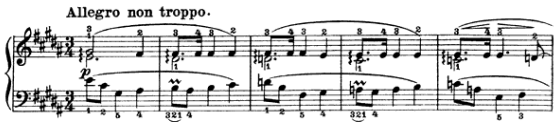


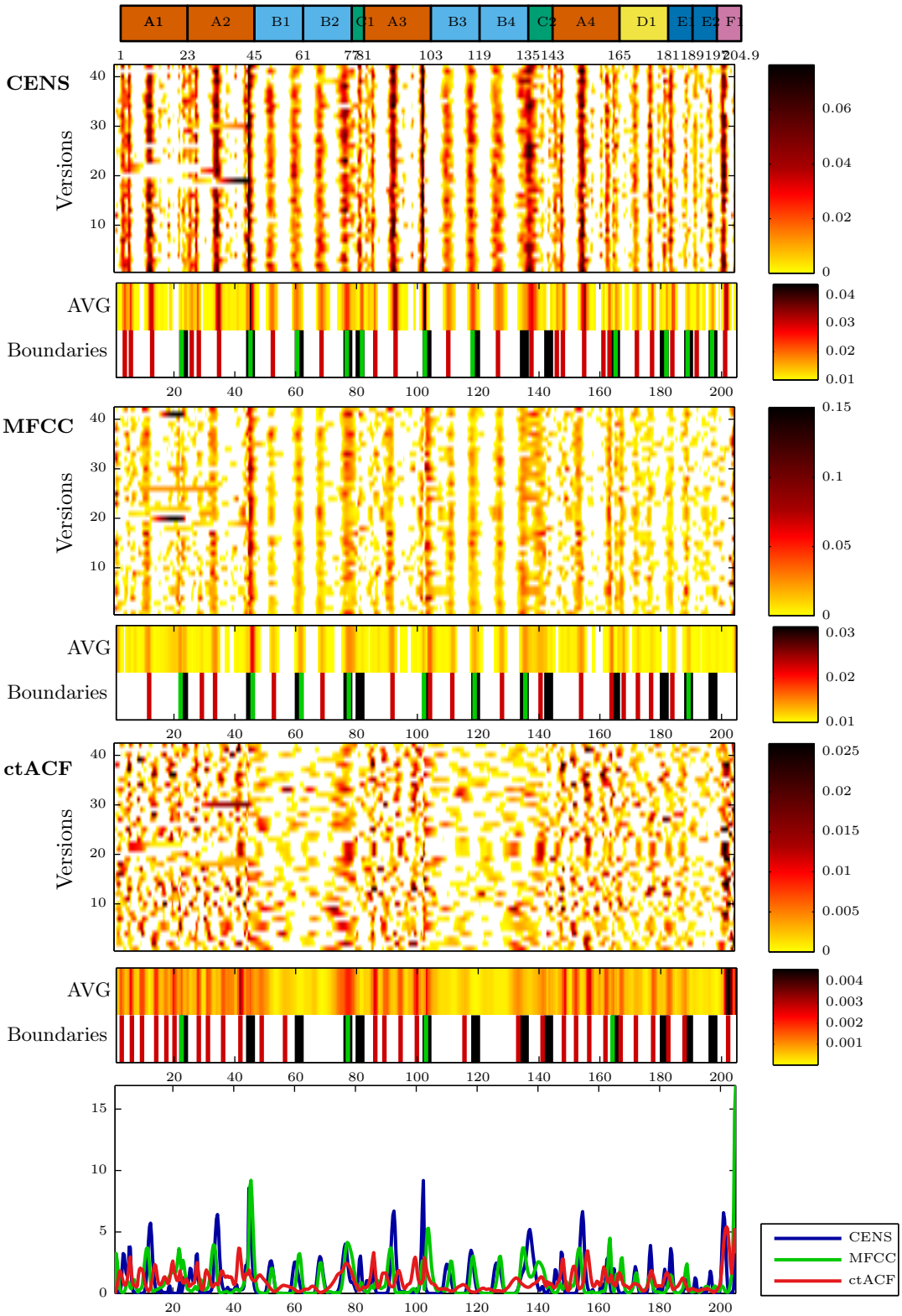
Mazurka 50 No. 3



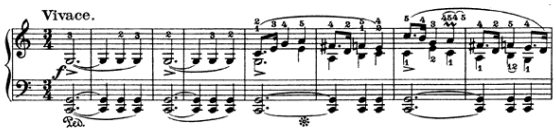


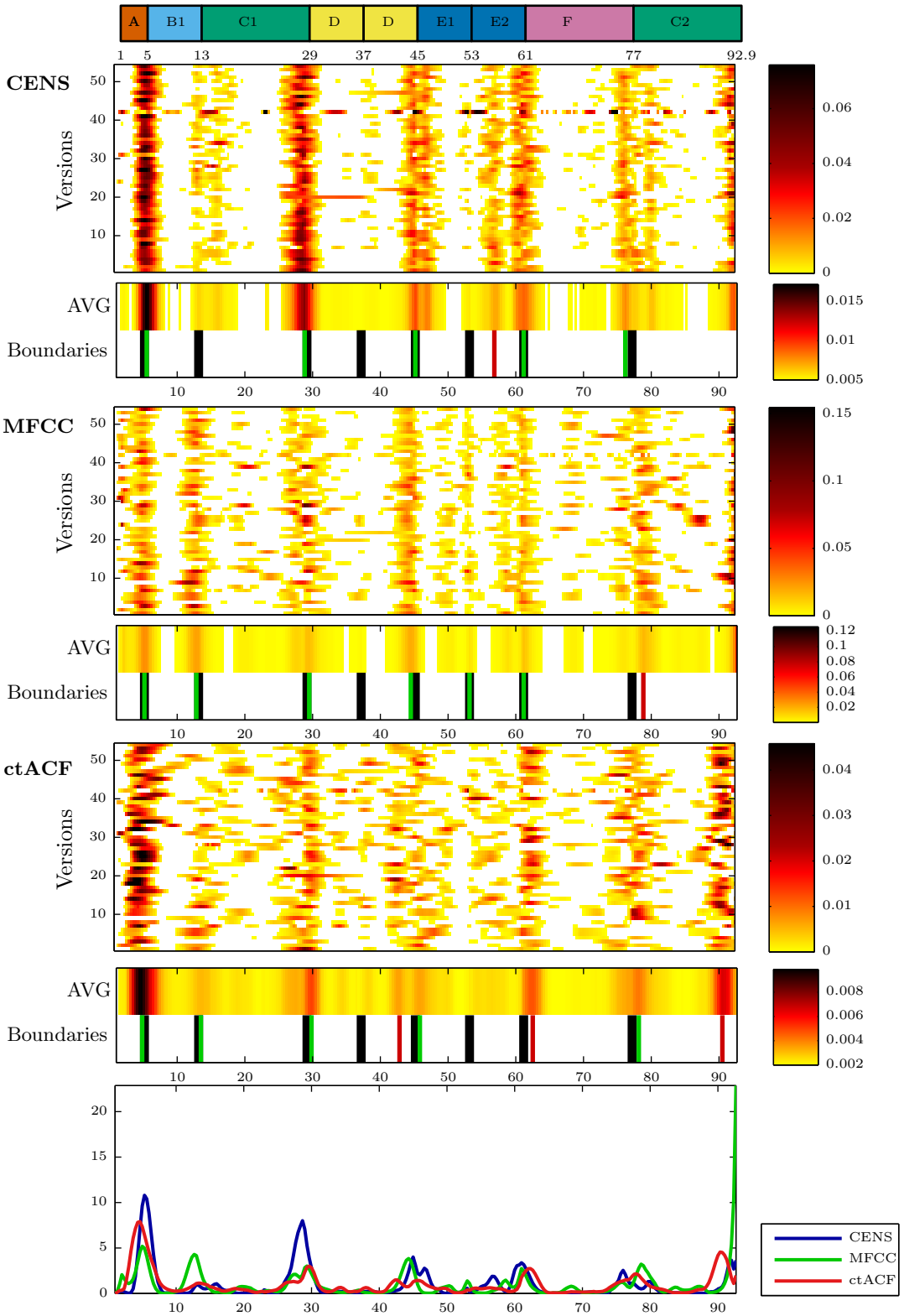
Mazurka 56 No. 1



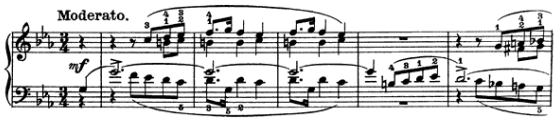


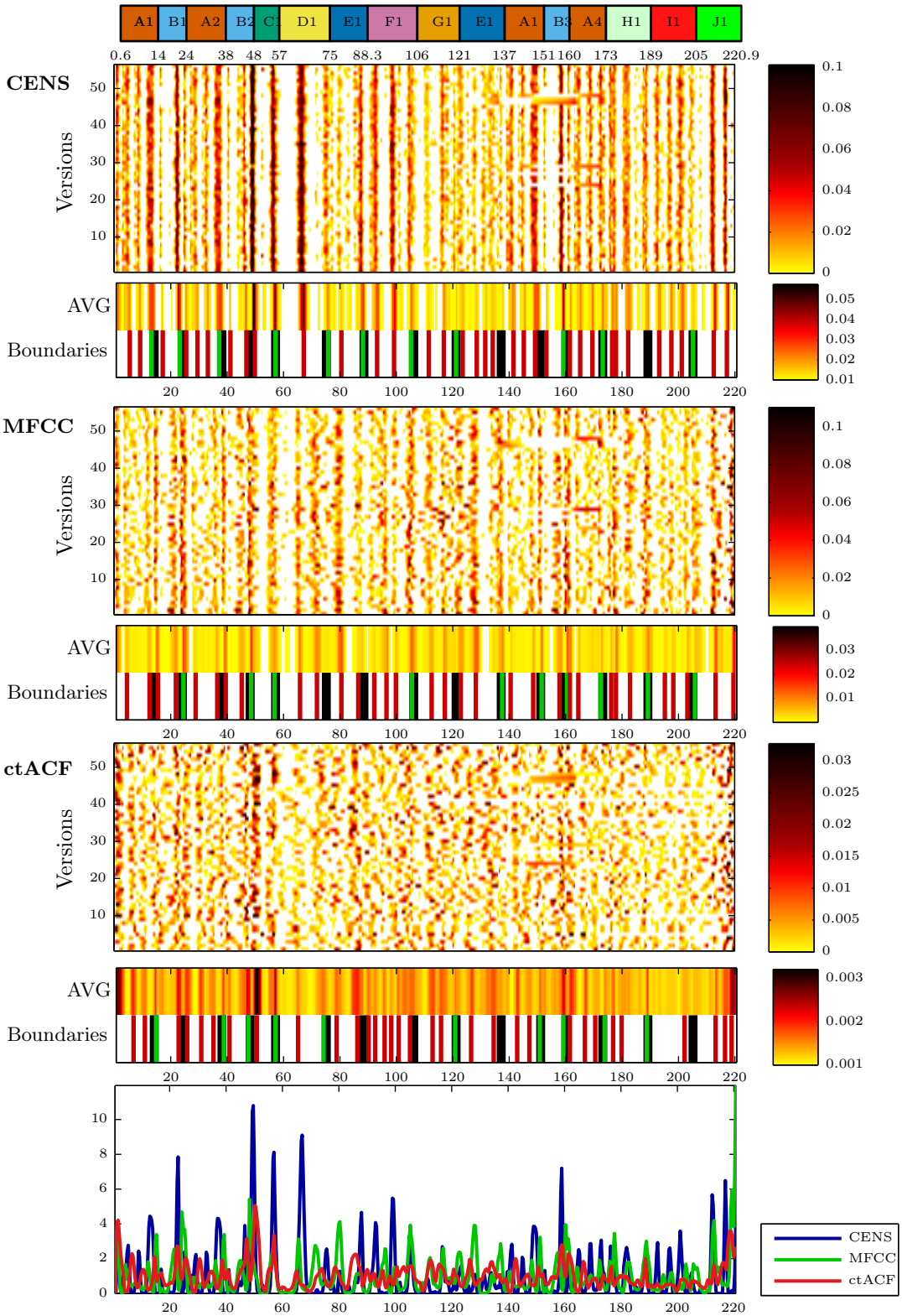
Mazurka 56 No. 2



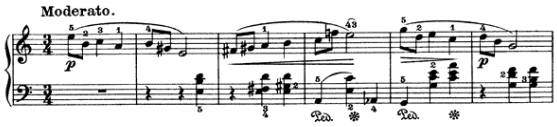


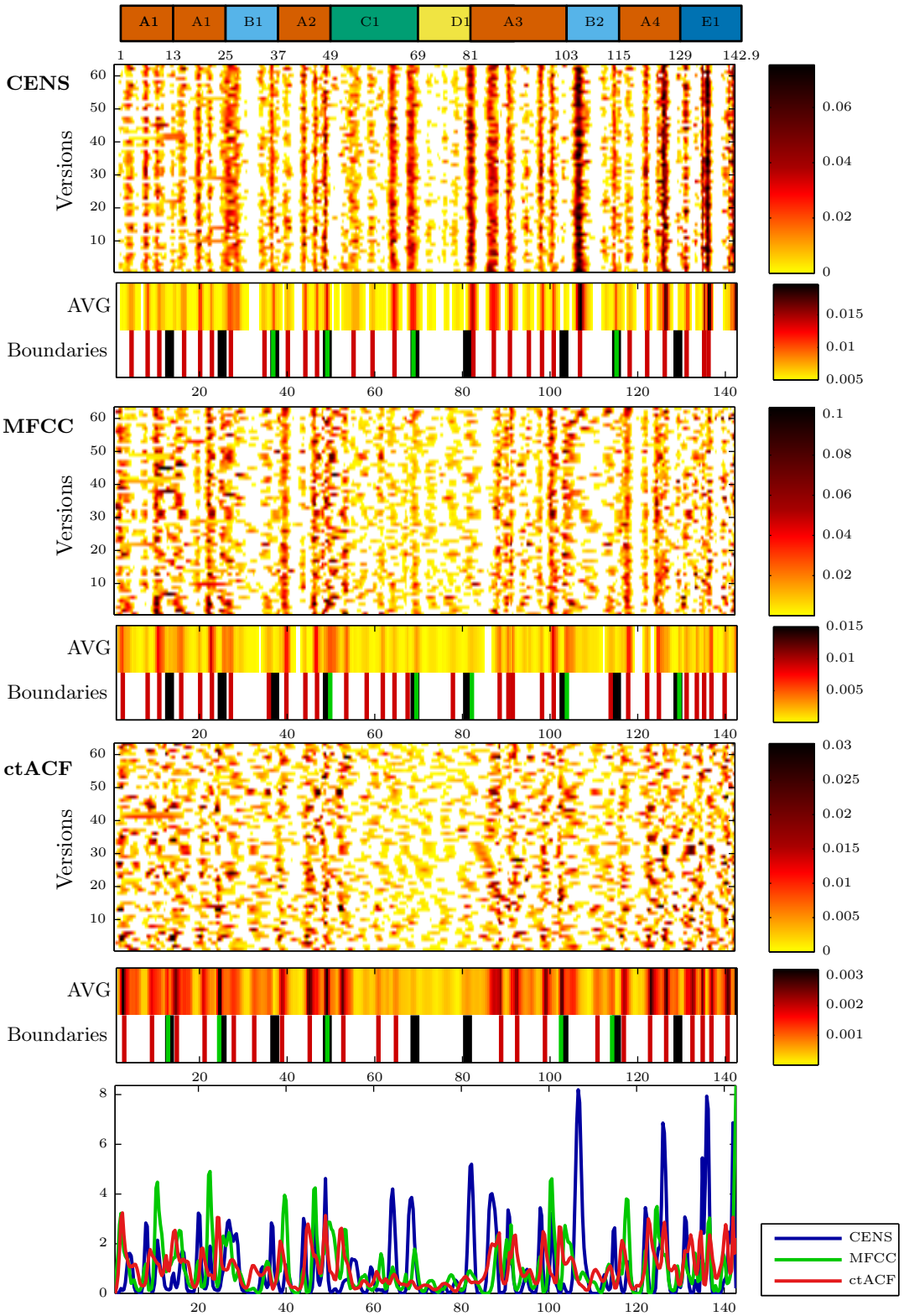
Mazurka 56 No. 3



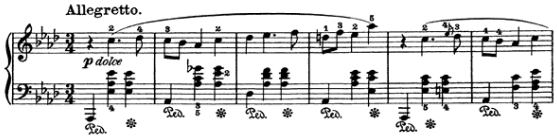


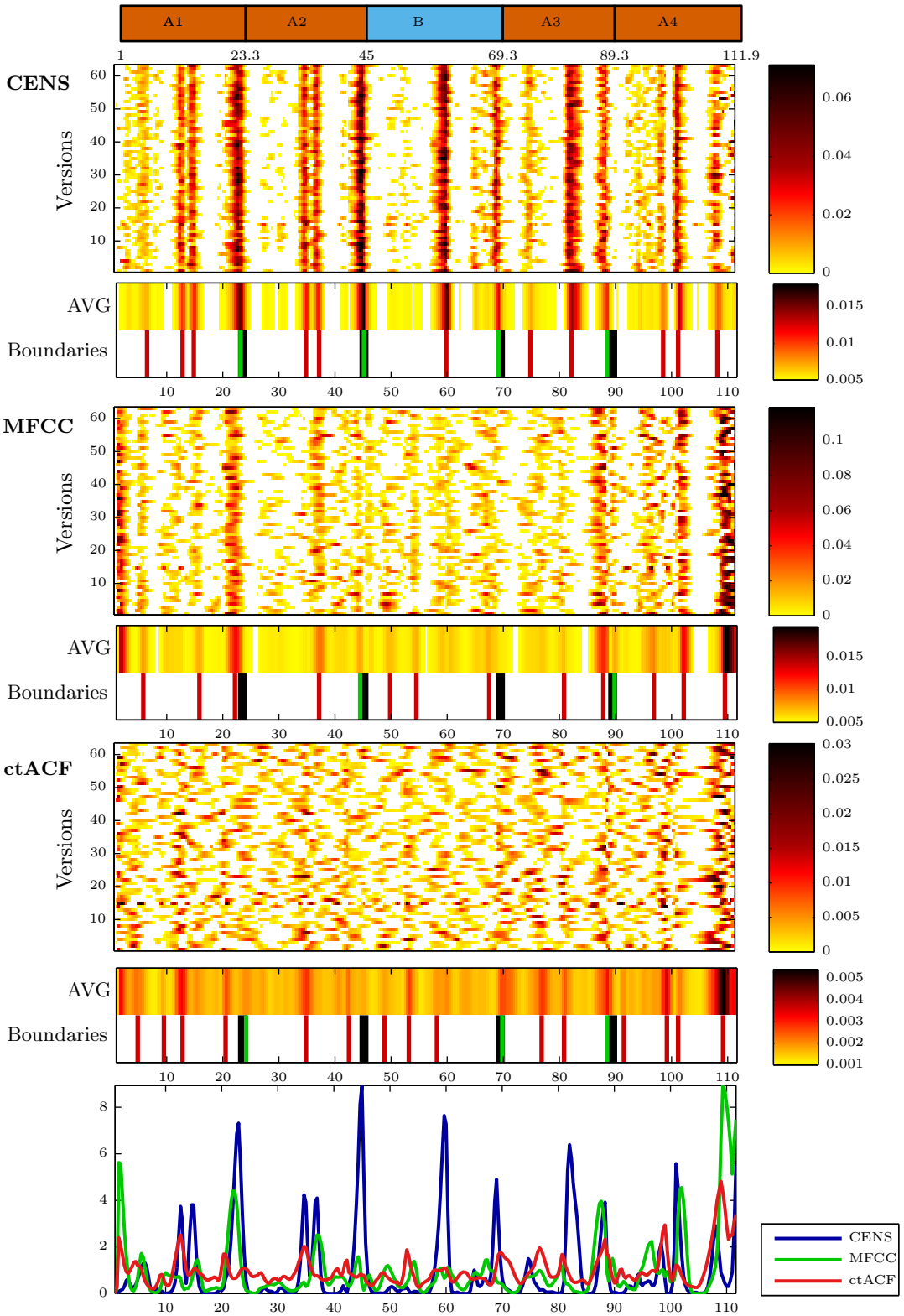
Mazurka 59 No. 1



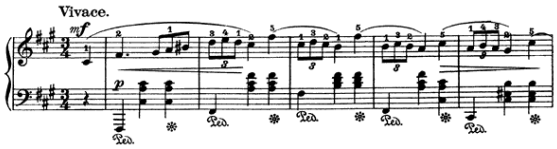


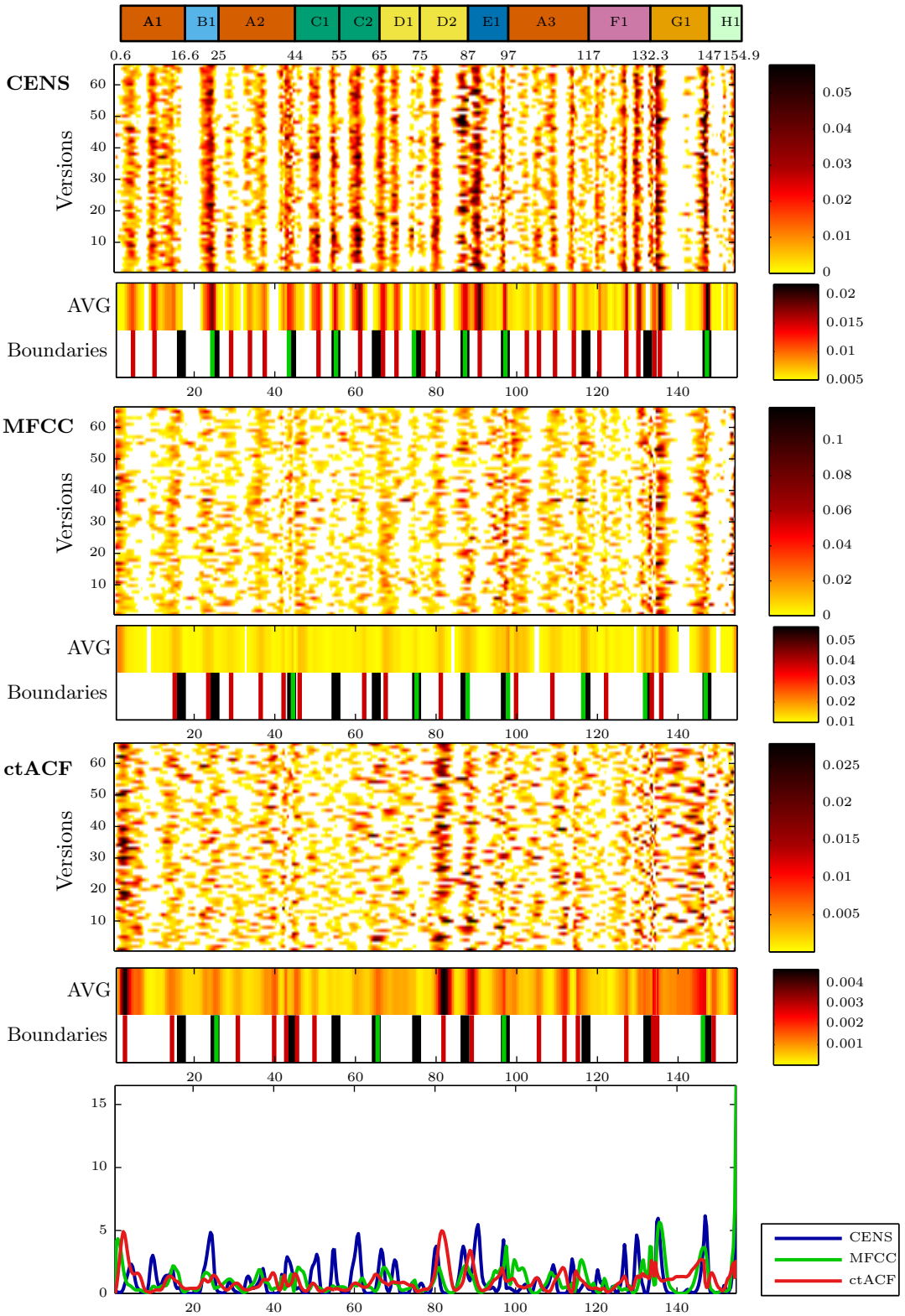
Mazurka 59 No. 2



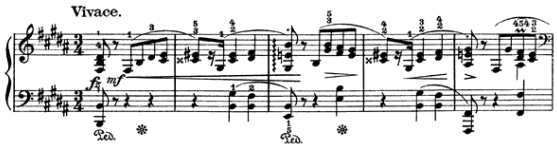


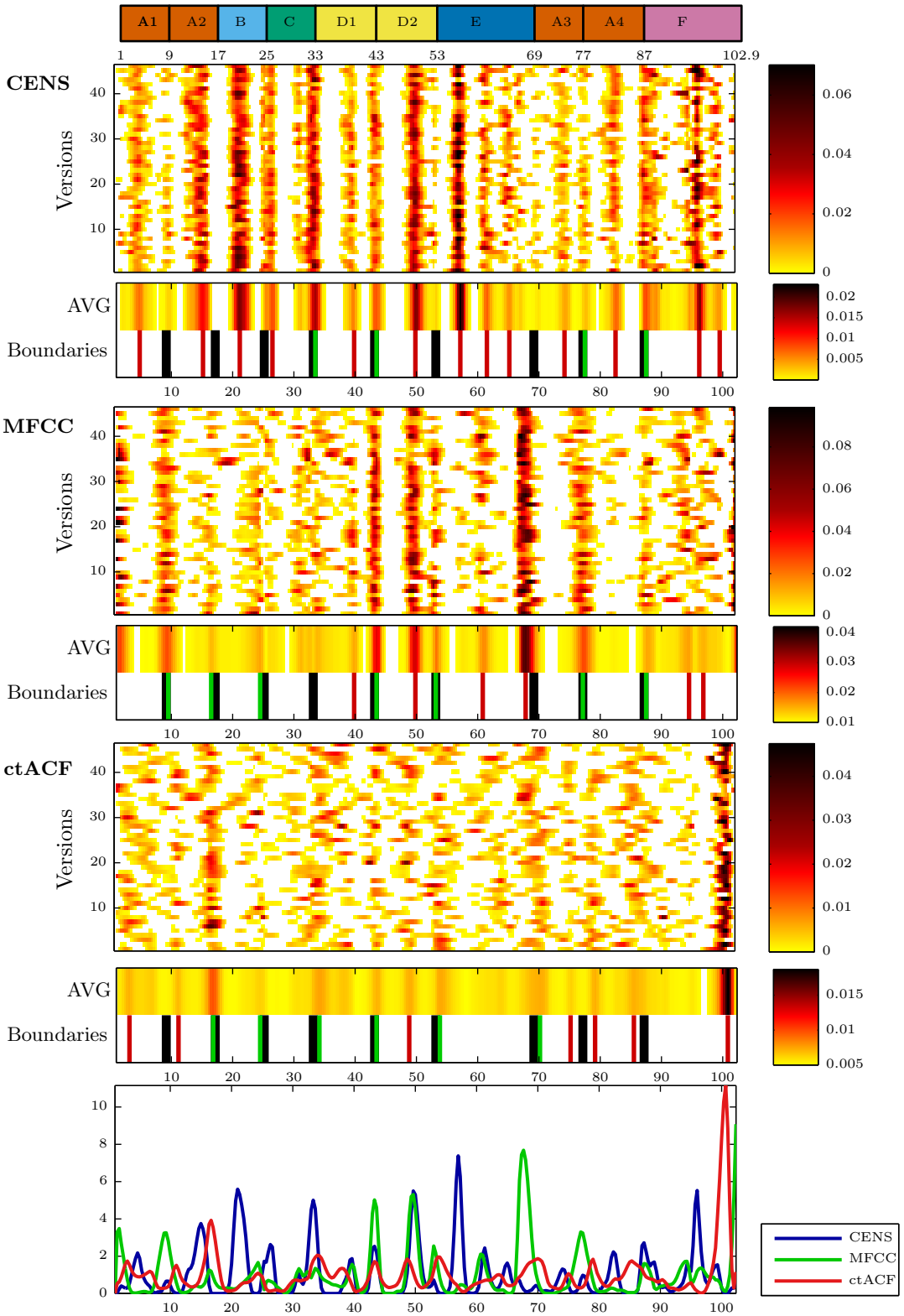
Mazurka 59 No. 3



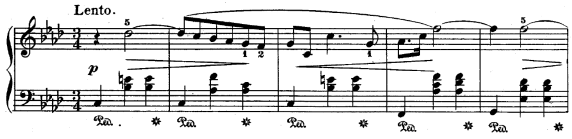


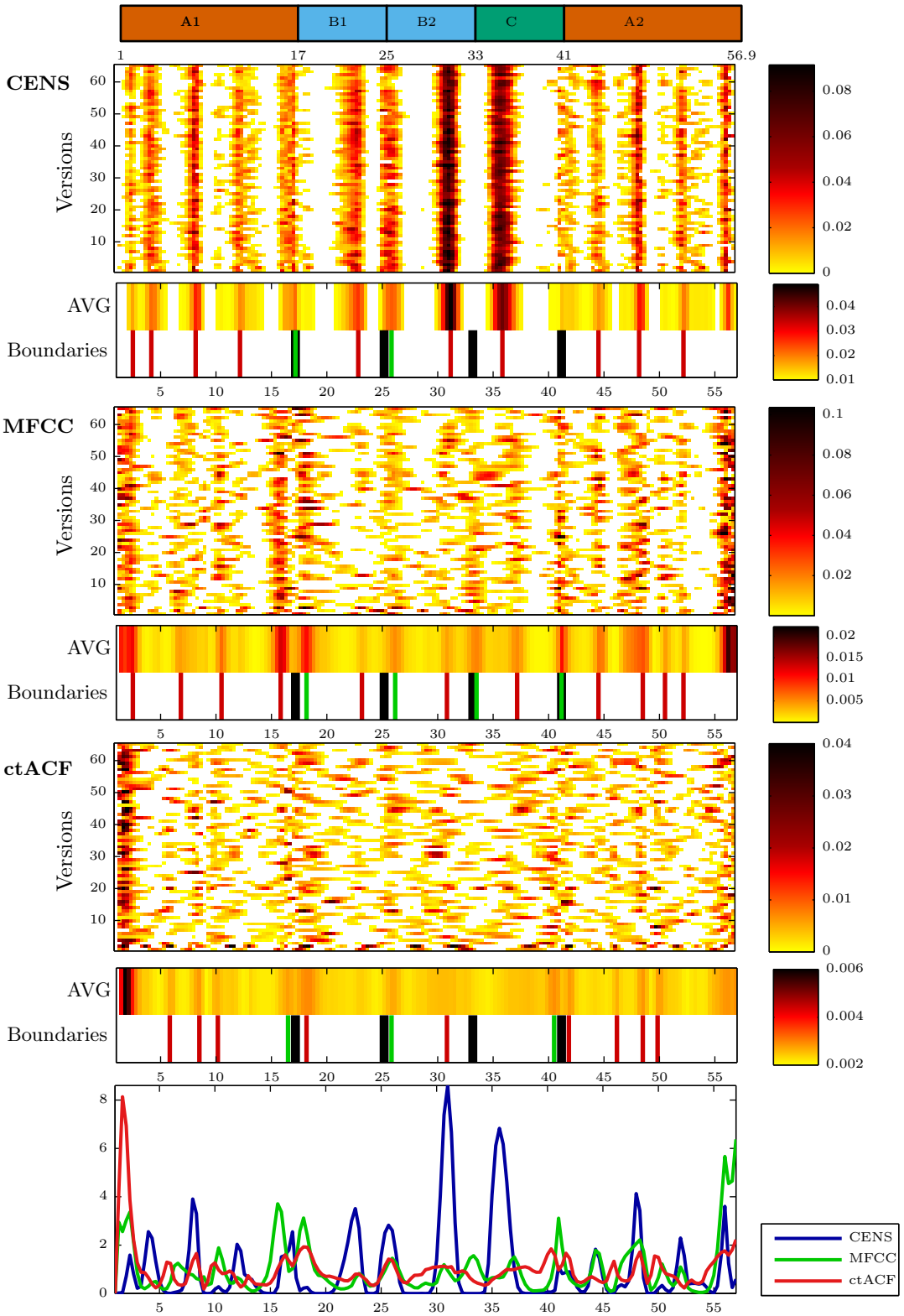
Mazurka 63 No. 1



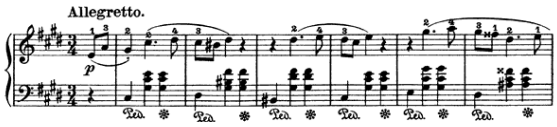


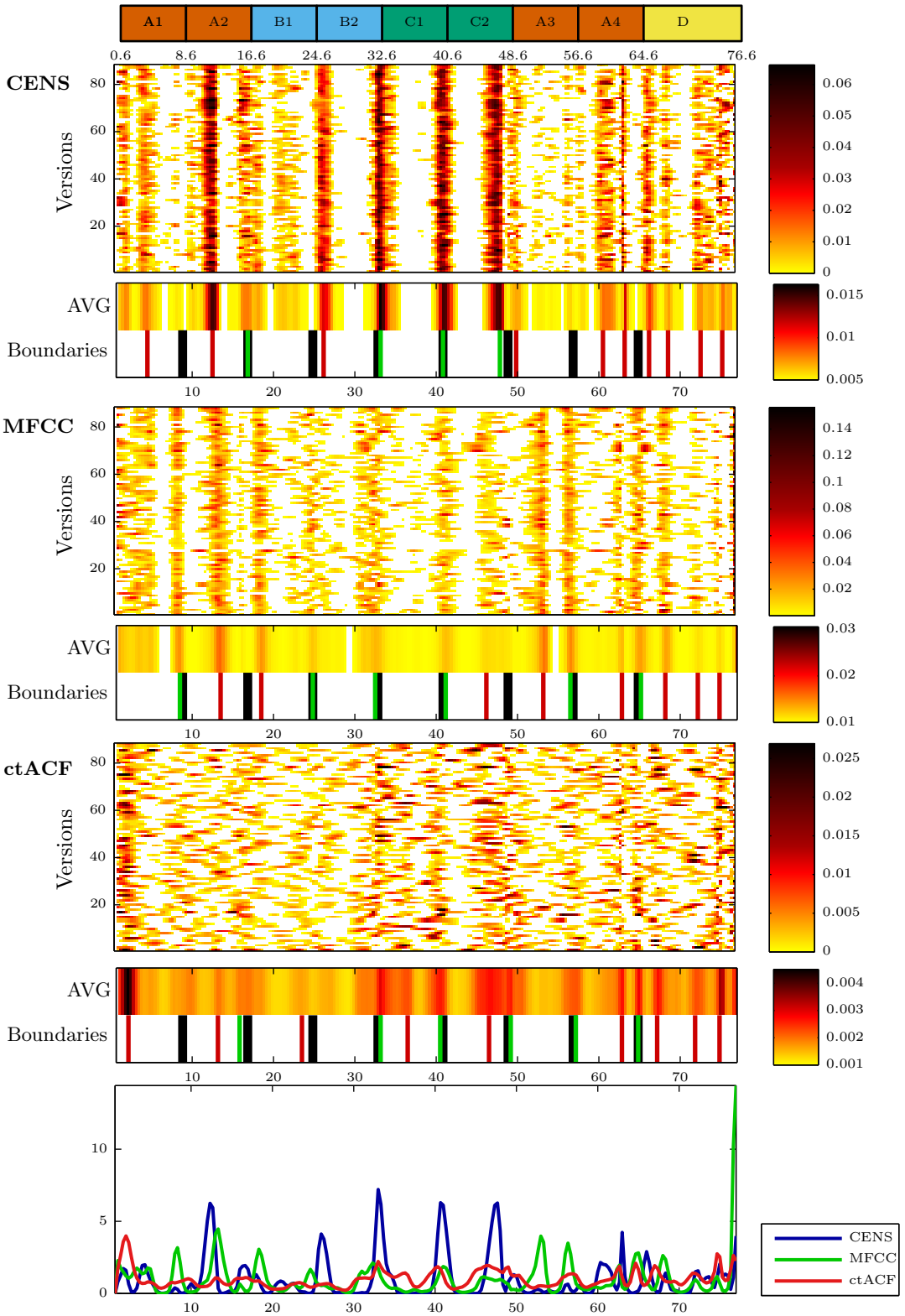
Mazurka 63 No. 2



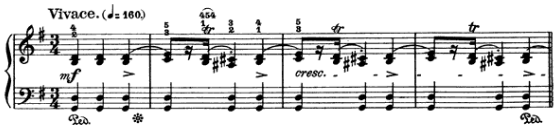


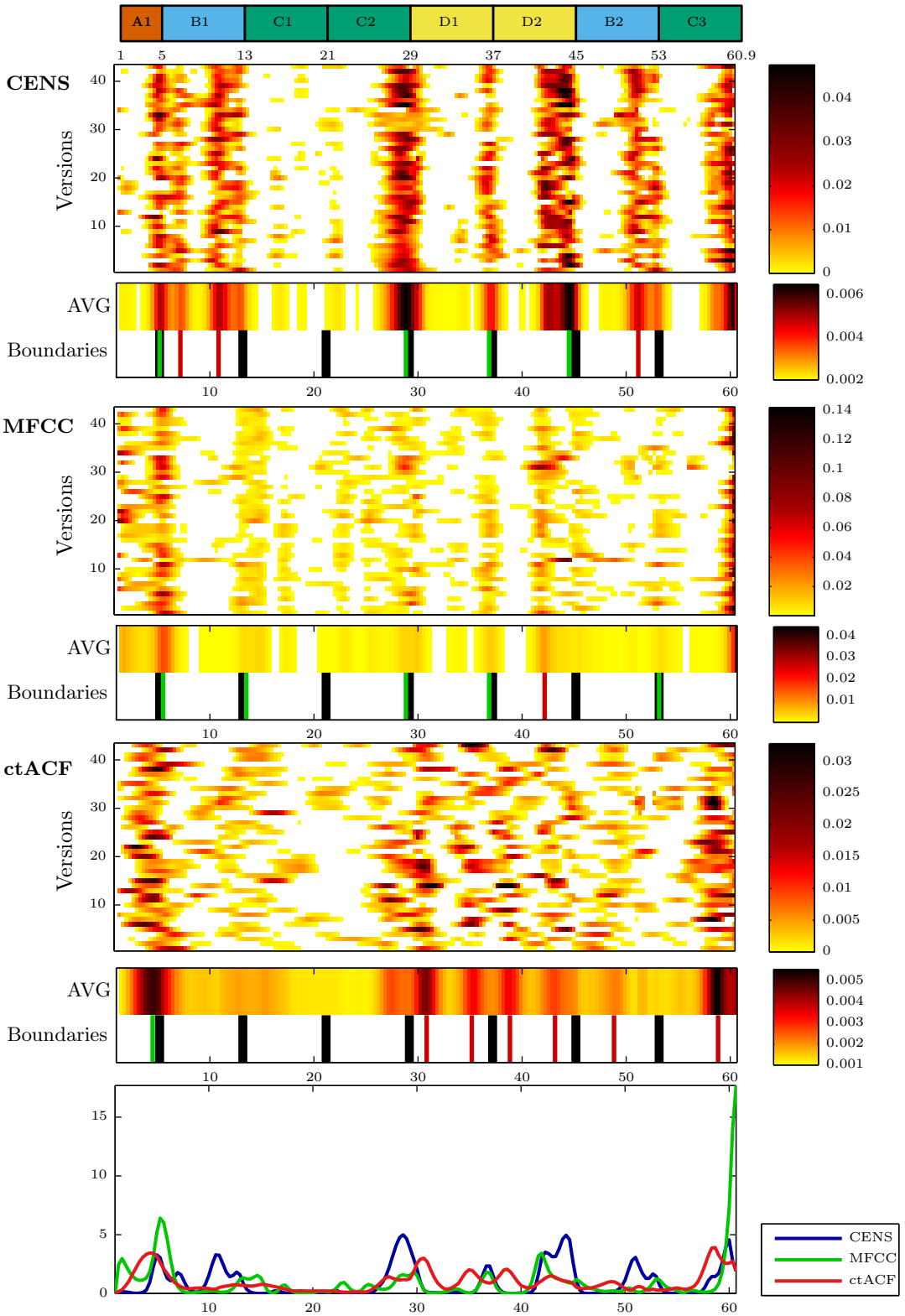
Mazurka 63 No. 3



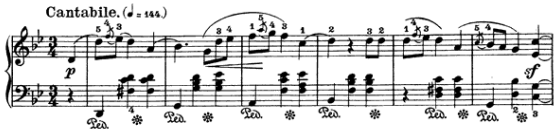


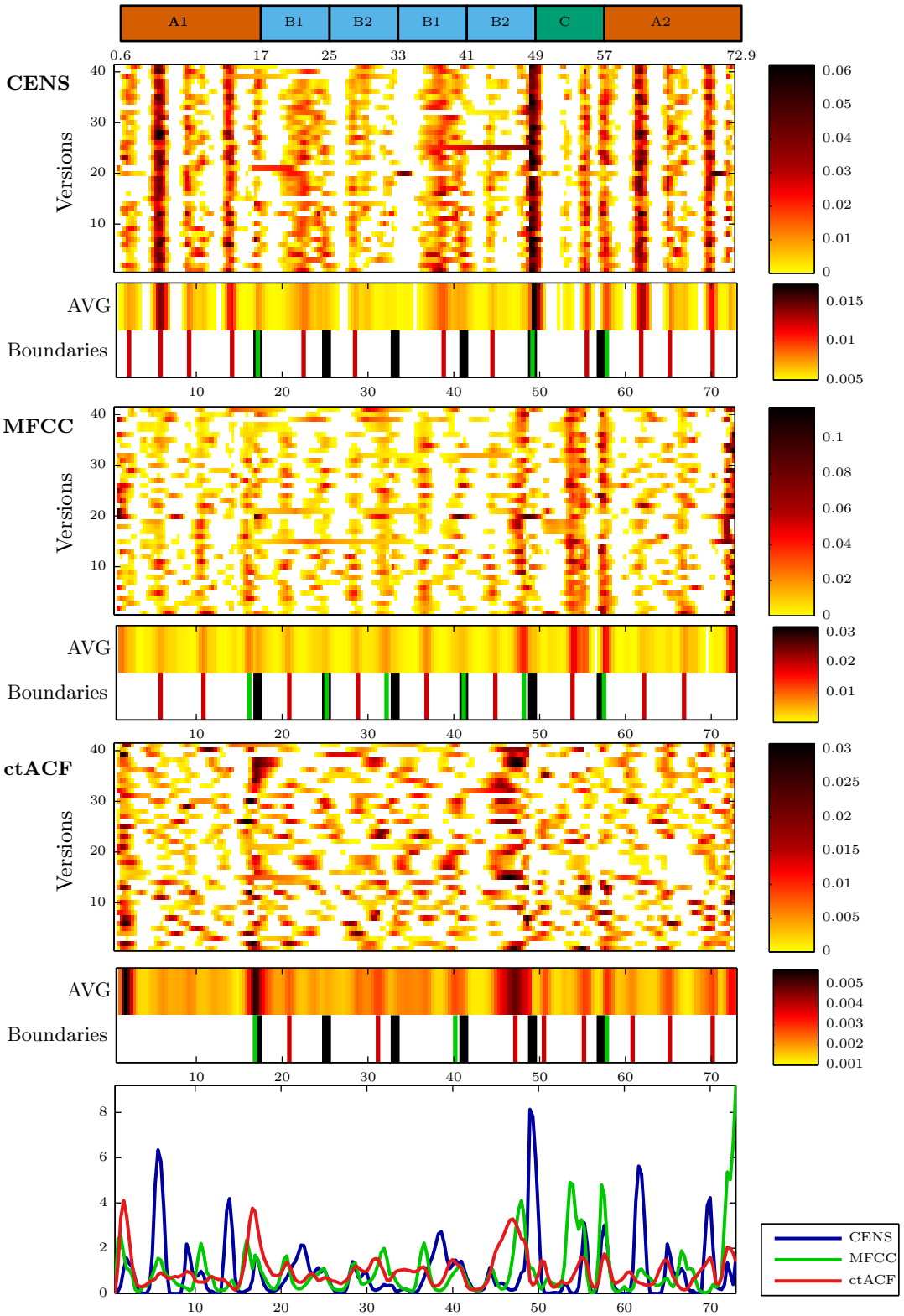
Mazurka 67 No. 1



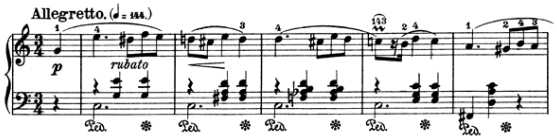


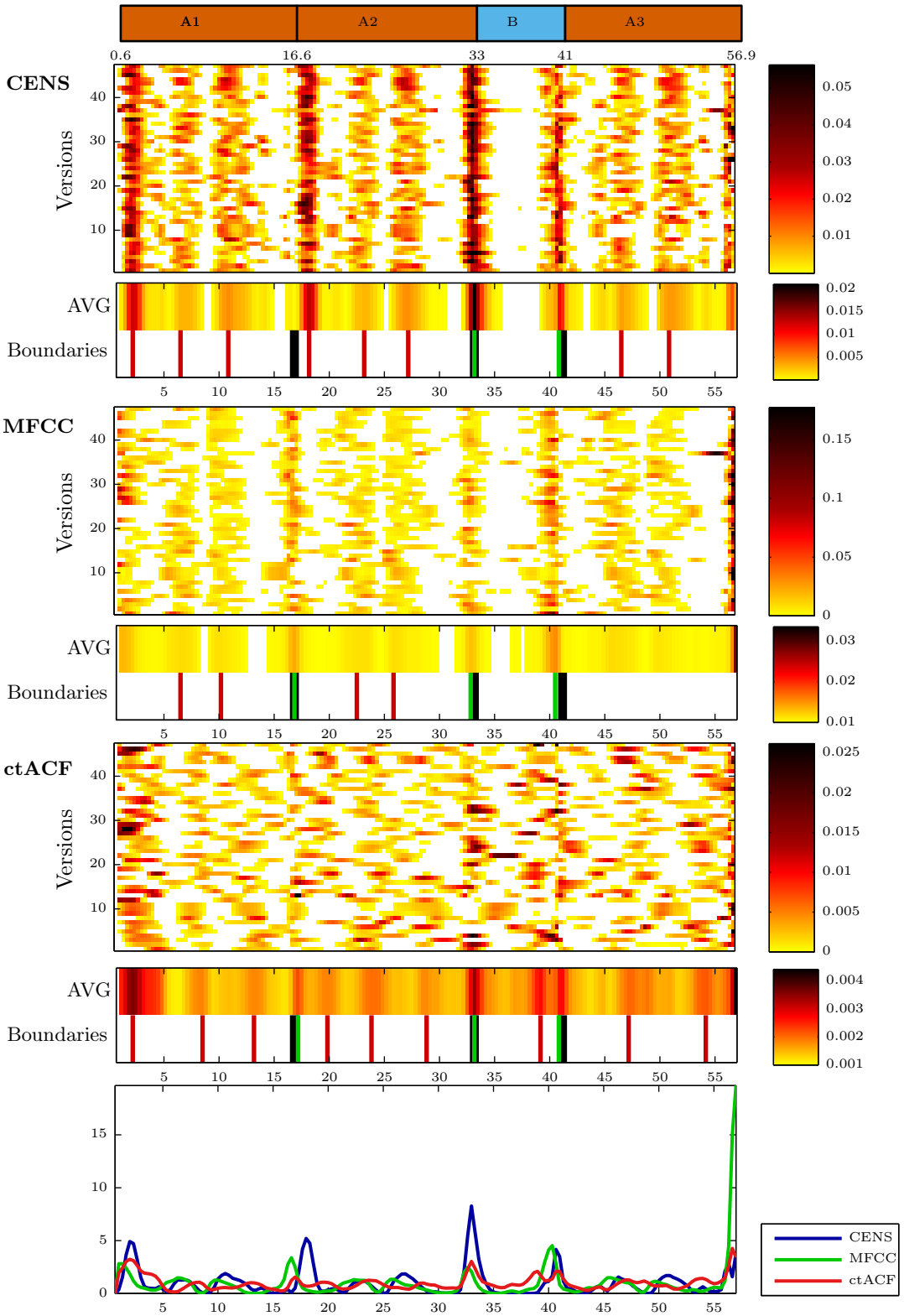
Mazurka 67 No. 2



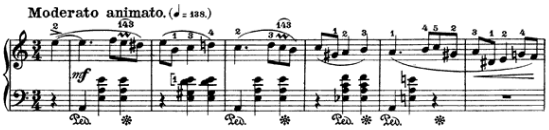


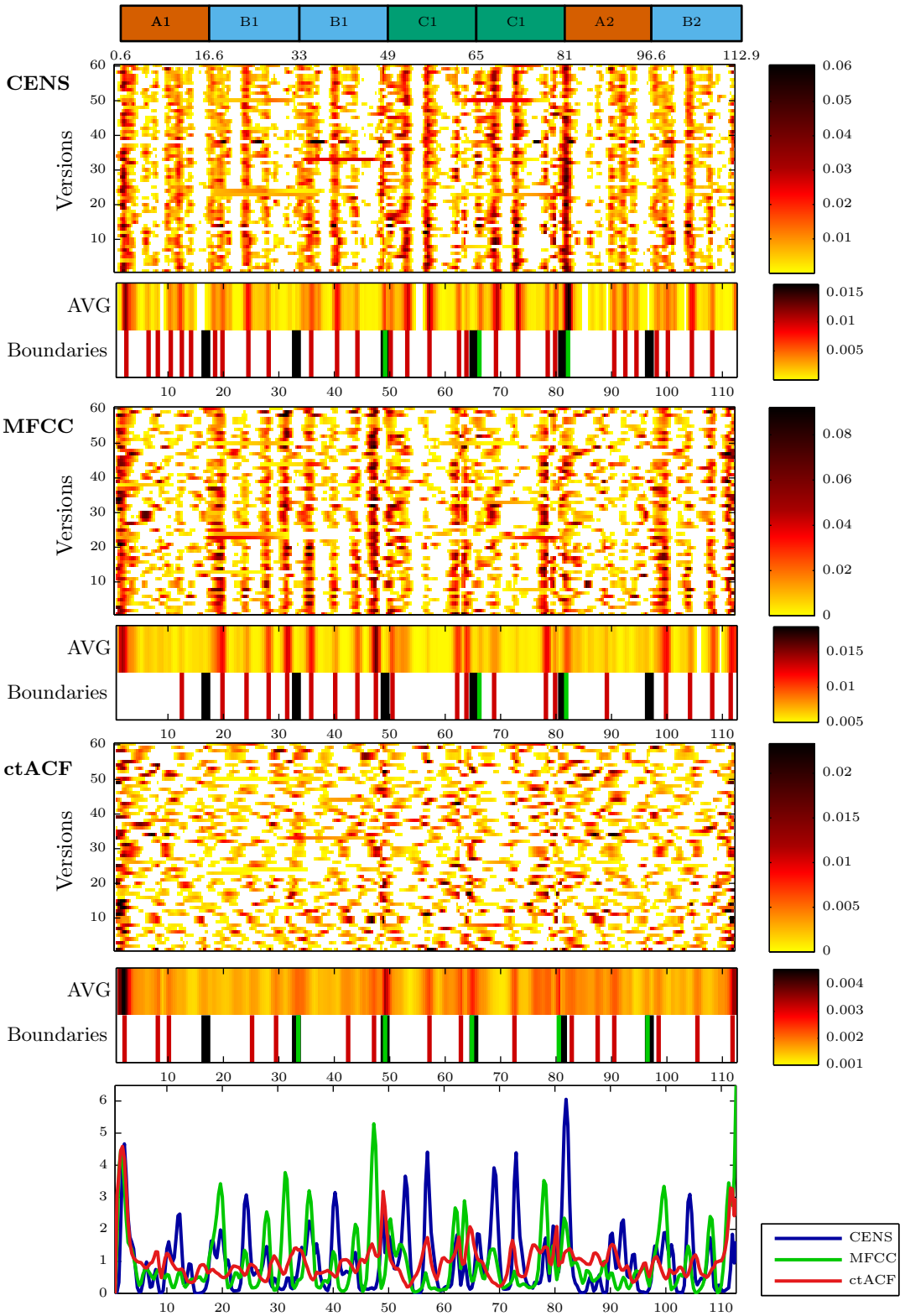
Mazurka 67 No. 3



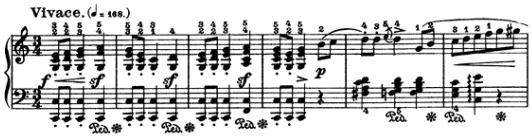


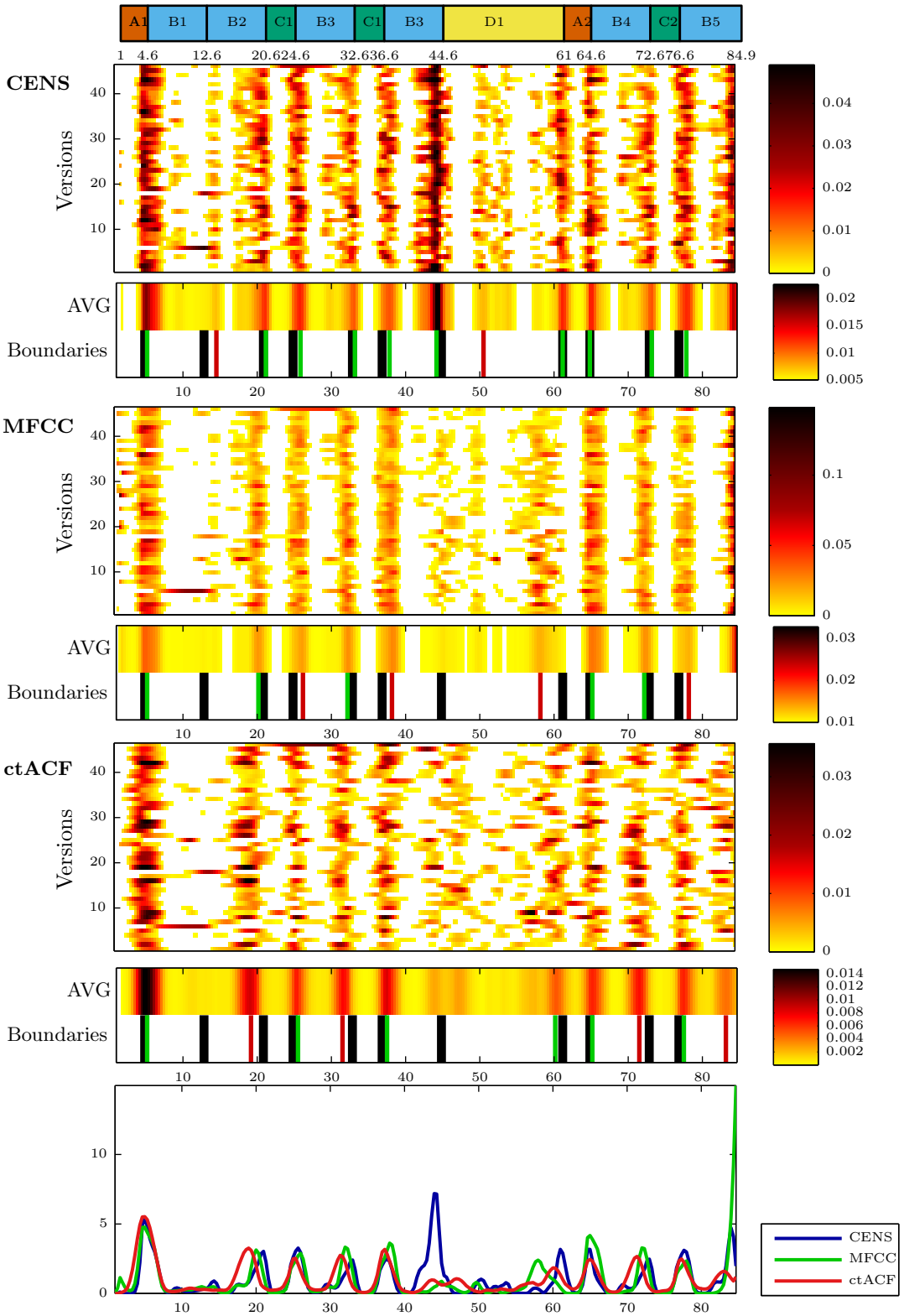
Mazurka 67 No. 4



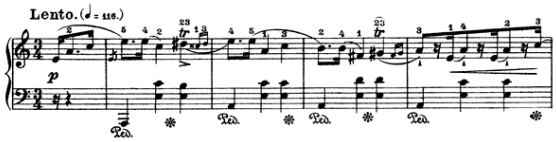


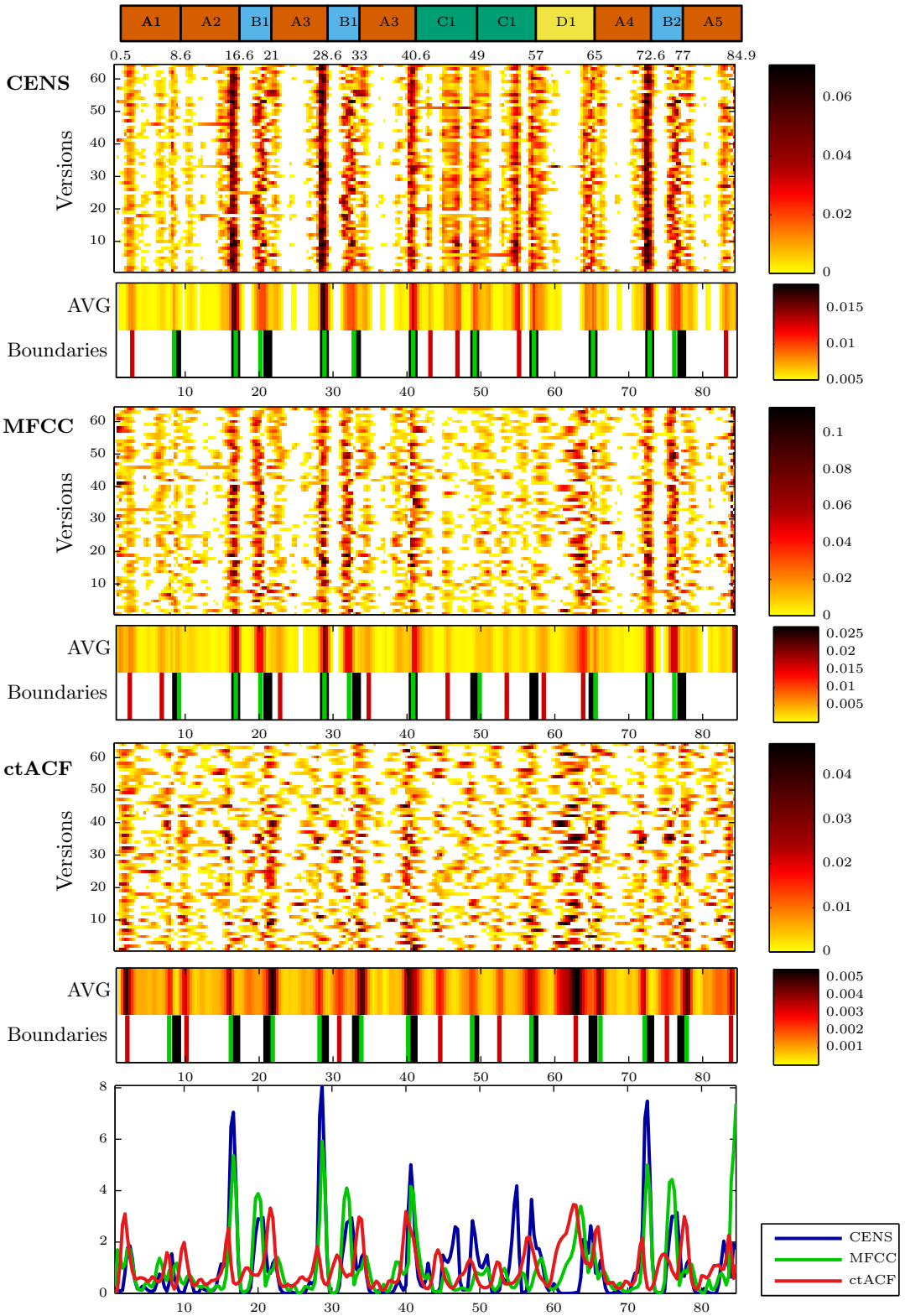
Mazurka 68 No. 1



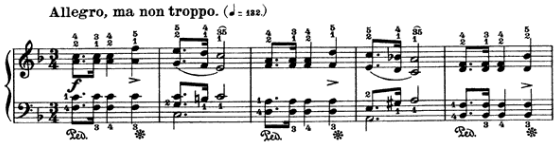


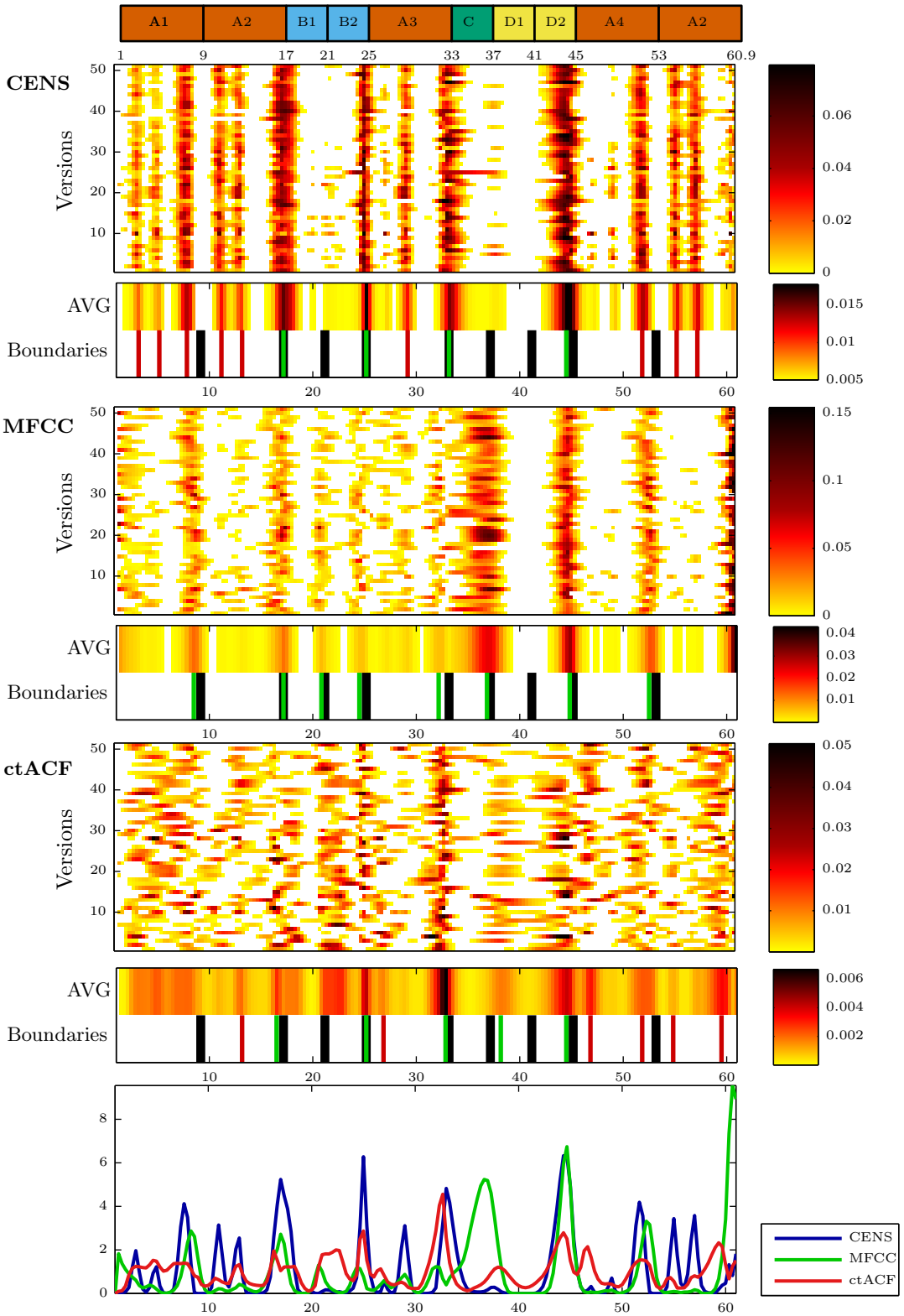
Mazurka 68 No. 2



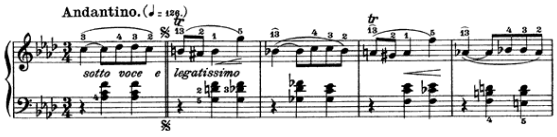


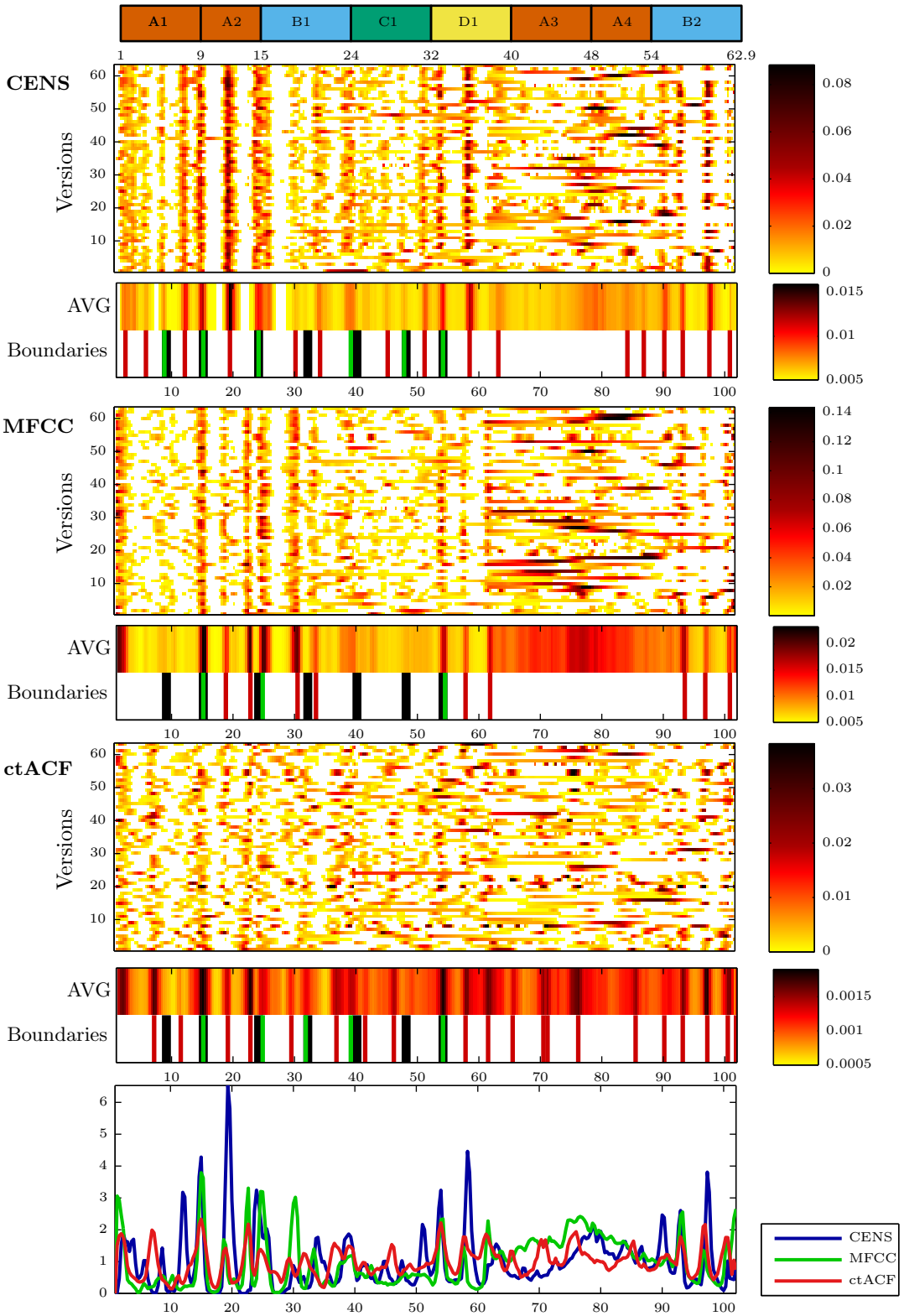
Mazurka 68 No. 3





Mazurka 68 No. 4





Bibliography

[1] J.-J. Aucouturier and F. Pachet, *Improving timbre similarity: How high's the sky*, Journal of Negative Results in Speech and Audio Sciences, 1 (2004).

[2] M. Cooper and J. Foote, *Automatic music summarization via similarity analysis*, in Proceedings of the International Conference on Music Information Retrieval (ISMIR), Paris, France, 2002.

[3] S. Ewert, *Effiziente Methoden zur hochauflösenden Musiksynchronisation*, master's thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, Institut fr Informatik III, 2007.

[4] S. Ewert and M. Müller, *Refinement strategies for music synchronization*, in Proceedings of the 5th International Symposium on Computer Music Modeling and Retrieval (CMMR), vol. 5493 of Lecture Notes in Computer Science, Copenhagen, Denmark, May 2008, pp. 147–165.

[5] S. Ewert, M. Müller, and P. Grosche, *High resolution audio synchronization using chroma onset features*, in Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Taipei, Taiwan, Apr. 2009, pp. 1869–1872.

[6] J. Foote, *Automatic audio segmentation using a measure of audio novelty*, in Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), New York, NY, USA, 2000, pp. 452–455.

[7] J. T. Foote and M. L. Cooper, *Media segmentation using self-similarity decomposition*, Storage and Retrieval for Media Databases, 5021 (2003), pp. 167–175.

[8] C. for the History and A. of Recorded Music, *Mazurka project*. Online.

[9] T. Fujishima, *Realtime chord recognition of musical sound: a system using common lisp music*, in Computer Music Conference (ICMC), 1999, pp. 464–467.

[10] P. Grosche and M. Müller, *Tempogram Toolbox: MATLAB tempo and pulse analysis of music recordings*, in 12th International Conference on Music Information Retrieval (ISMIR, late-breaking contribution), Miami, USA, 2011.

[11] P. Grosche, M. Müller, and F. Kurth, *Cyclic tempogram – a mid-level tempo representation for music signals*, in Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Dallas, Texas, USA, Mar. 2010, pp. 5522 – 5525.

[12] P. Grosche, M. Müller, and C. S. Sapp, *What makes beat tracking difficult? A case study on Chopin Mazurkas*, in Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR), Utrecht, Netherlands, 2010, pp. 649–654.

[13] M. Hein, *Machine Learning*, Lecture Notes, February 2009.

[14] D. Huron, *Sweet Anticipation: Music and the Psychology of Expectation (Bradford Books)*, The MIT Press, 1 ed., May 2006.

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

[16] V. Konz, M. Müller, and S. Ewert, *A multi-perspective evaluation framework for chord recognition*, in Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR), Utrecht, Netherlands, 2010.

[17] B. Logan, *Mel frequency cepstral coefficients for music modeling*, in Proceedings of the International Symposium on Music Information Retrieval (ISMIR), Plymouth, Massachusetts, 2000.

[18] C. J. Moore, ed., *Hearing, Handbook of Perception and Cognition*, Academic Press, 2 ed., 1995.

[19] M. Müller, *Information Retrieval for Music and Motion*, Springer Verlag, 2007.

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

[21] M. Müller, V. Konz, N. Jiang, and Z. Zuo, *A multi-perspective user interface for music signal analysis*, in Proceedings of the International Computer Music Conference (ICMC), 2011, pp. 205–211.

[22] N. Orio and D. Schwarz, *Alignment of monophonic and polyphonic music to a score*, in in Proceedings of the ICMC, 2001, pp. 155–158.

[23] E. Pampalk, *A matlab toolbox to compute similarity from audio.*, in Proceedings of the 2004 International Conference on Music Information Retrieval, New York, NY, USA, 2004, pp. 254–257.

[24] J. Paulus, M. Müller, and A. Klapuri, *Audio-based music structure analysis*, in Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR), Utrecht, Netherlands, 2010, pp. 625–636.

[25] C. S. Sapp, *Hybrid numeric/rank similarity metrics*, in Proceedings of the International Conference on Music Information Retrieval (ISMIR), Philadelphia, USA, 2008, pp. 501–506.

[26] J. B. L. Smith, *A comparison and evaluation of approaches to the automatic formal analysis of musical audio*, master's thesis, Music Technology Area, Department of Music Research, Schulich School of Music, McGill University, 2010.

[27] S. Smith, *The scientist and engineer's guide to digital signal processing*, California Technical Pub., 1997.

[28] P. von Styp-Rekowsky, *Towards time-adaptive feature design in music signal processing*, master's thesis, Saarland University, Department of Computer Science, 2011.