D2.1.2 Semantic descriptors bottom-up computation module

Abstract

SIMAC's Work Package 2 (WP2) is mostly concerned with the bottom-up estimation of semantic descriptors and the generation of metadata for semantic analysis. Within this context, deliverable D2.1.1 comprises the low-level signal processing tools that support the calculation of descriptors (which is the aim of D2.1.2).

In this document we explain in detail the scope of deliverable D2.1.2 and provide documentation for a partial set of the tools being developed under this deliverable.

Version 3.0

Date: 18-July-2005

Editors: Juan Bello

Contributors: N. Chétry, M. Davies, S. Dixon, E. Gomez, F. Gouyon, C. Harte, P. Herrera, C. Landone, A. Nesbit, B. S. Ong, E. Pampalk, J. Pickens, V. Sandvold, S. Streich and X. Wen

Reviewers: M. Sandler, E. Pampalk, D. Wallace, C. Landone and O. Celma

Contents

Chapter 1

Introduction

Bottom-up semantic descriptors, or mid-level representations, of music are measures that can be computed directly from audio signals using a combination of signal processing, machine learning and musical knowledge. They seek to emphasise the musical attributes of audio signals (e.g. chords, rhythm, instrumentation), attaining higher levels of semantic complexity than low-level features (e.g. spectral coefficients, MFCC, etc), but without being bounded by the constraints imposed by the rules of music notation. Their appeal resides in their ability to provide a musically-meaningful description of audio signals that can be used for music similarity applications, such as retrieval, segmentation, classification and browsing in musical collections.

Previous attempts to model music from complex audio signals concentrate mostly on the use of low-level feature sets, that can be loosely associated to textural and rhythmic attributes in the signal [AP02a, Yan02].

Alternatively, the SIMAC approach proposes that we can successfully characterise music according to one or more "musical dimensions" (i.e. rhythm, harmony, melody, timbre) by adding higher-level descriptors to a low-level feature set. An example of this approach, originally proposed by Dixon et al. [DGW04b], and presented in Chapter 2, demonstrates success for classification according to rhythmic patterns. This is an example of or view that the use of knowledge, whether musical, cognitive or user-oriented, can overcome some of the shortcomings of the purely data-driven approach to music retrieval. We are interested on characterising music signals beyond the audio (as opposed to common fingerprinting approaches), by encoding knowledge into our representations that is closer to the generative or the user level.

We are extensively exploring this research trend in SIMAC's work packages 2 and 3, as exemplified by work reported in deliverable D3.4.1, where mid-level descriptors, such as the chromagram have been used to develop measures of complexity (closely related to chord/harmonic complexity). Although results are preliminary, they suggest that the "glass ceiling" pointed out in previous data-driven approaches reported in D3.1.1 and D.3.2.1, might be higher than originally anticipated.

1.1 Overview of this report

The remainder of this document is organised as follows. The rest of this introduction is concerned with the details of WP2 and the proposed description of this deliverable, including its testing and evaluation.

We then introduce our description scheme as a function of high-level musical dimensions, thus presenting several example approaches, developed under SIMAC, for the description of rhythm (Chapter 2), harmony (Chapter 3), timbre and instrumentation (Chapter 4), long-term structure (Chapter 6), complexity (Chapter 7) and similarity (Chapter 8) in music signals. During all chapters we discuss both published research and work in progress, with an emphasis on novel approaches to expand our musically-meaningful description of signals.

In the Appendices A, B and C, we propose manual and semi-automatic methodologies for generating annotated data, thus addressing one of the major difficulties for the design and evaluation of our semantic description scheme.

1.2 About Work Package 2:

Work Package 2 (WP2), entitled Analysis and Processing of music signals, is coordinated by QMUL and it is 24 months in duration. It is mostly a research and development package whose main goal is the design and implementation of algorithmic tools for the bottom-up generation of semantic descriptors from musical signals and for the processing of music contents for navigation and visualisation purposes.

Workpackage goals:

- To provide the low-level signal processing tools for analysing music signals from a bottom-up perspective
- To provide the algorithms for the bottom-up estimation of semantic descriptors.
- To provide the content processing tools for the enhanced playback of songs according to semantic criteria
- To provide semantic markup and metadata from the analysis.

1.3 WP2 deliverables

There are three deliverables in WP2 (final delivery months in brackets):

- D2.1.1 Front-end signal processing and low-level descriptors computation module (Month 12)
- **D2.1.2 Semantic descriptors bottom-up computation module (Month 18)**
- D2.3.1 Music content processing tools (Month 24)

1.4 D2.1.2 Semantic descriptors bottom-up computation module

- Deliverable name: Semantic descriptors bottom-up computation module
- Type: Official
- Lead participant: Queen Mary University of London (QMUL)
- Purpose: This deliverable supplies algorithms for the bottom-up computation of semantic descriptors from music signals. Most generated descriptors will have direct applicability to the prototypes being developed as part of WP5 and/or will be used during WP3 to create robust music similarity measures or to generate new semantic descriptors by finding correlations between these bottom-up features and user-generated metadata. This deliverable covers seven groups of tasks: rhythm, harmony, timbre and instrumentation, acoustic, segmentation, complexity and similarity.

  - Rhythm-related descriptors identify temporal periodicities, patterns and clusters of temporal events, which characterise the rhythm of a musical piece in a digital recording.
  - Harmony-related descriptors characterise the harmony and tonality of a musical piece. Also related to the melodic content.
  - Instrumentation/timbre descriptors are intended to characterise the quality (and possibly the quantity) of the instrumentation in musical recording. Also aim to describe the timbral content in a signal as a function of abstract attributes related to the way users interact with music contents.
  - Acoustic descriptors refer to characteristics of the sound related to acoustic properties of the recording and are mostly associated to the intensity and the spatial attributes of the signal.
  - Segmentation descriptors identify the long-term structure of a musical piece (e.g. verse-chorus-verse-bridge-chorus) by finding blocks of common attributes using mostly low-level feature representations of music signals. This can be used as prior information for other identification tasks, such as instrumentation and dynamics, and as corroborative evidence for rhythm and tempo analysis.
  - Complexity descriptors are associated to one or more musical dimensions, and characterise the effort or the amount of knowledge to be used by the listener to understand the information in the signal. They can also be seen as quantifying the nature (and size) of the "palette" used to compose the music within a song.
  - Music Similarity refers to those attributes that groups of songs (e.g. belonging to the same genre) have in common. Due to its nature, music similarity only has validity at collection level.

- This deliverable includes:
  - Matlab and C++ source code will be uploaded to the CVS server
  - A report, the library documentation (on the online description scheme) and user guidelines
  - Data test-set.
- Derivation: This deliverable relies on D.2.1 and on the annotated test DBs created in WP4 (D.4.3.1 and D.4.3.3). User requirements and feedback from WP4 (D.4.1.2 and D.4.1.3) will direct the research focus, so will do research outcomes of deliverables D3.1.1 and D3.2.1 Resulting algorithms will be used in WP3 (mostly D3.4.1) and WP5.
- Format and presentation:
  - Matlab and C++ source code will be uploaded to the CVS server.
  - The documentation will be written in English and it will be delivered in PDF and text formats.
- Quality criteria:

  - Does the library provides useful descriptors for the top-down approach to be implemented in WP3, and for the prototypes been developed in WP5?
  - Is it multiplatform? Does the C++ compile under windows and linux ?
  - Does it execute the needed functionality in the required time?
  - Has it been verified by the other involved partners?

- Quality method:
  - Feedback to design from WP3 and WP5
  - Compile in both development environments.
  - Test it with the test-set
  - Verify the process time with-in specs
  - Forward to other partners for verification.

- Other involved participants: MD, OFAI and UPF
- Estimated effort: MM 42
- Nature: Software + documentation
- Dissemination level: Consortium Only
- Delivery dates: - 31/06/2005 - beta version - 1/03/2006 - final version
- Internal reviewers: Elias Pampalk (OFAI), David Wallace (MD), Christian Landone (QMUL), Xavier Amatriain (UPF)
- External review: Project Officer

1.5 Testing and quality criteria

1.5.1 Test methodology

A description of the testing procedure for the library of descriptors is detailed in the following:

- Direct review: Compare the algorithm implementation to bibliographic sources. This includes contrasting sources when more than one is available.
- Singularities handling and unit testing: Algorithms are tested against numerical singularities (e.g. 0/0, x/0, -x1/2, ln(-x)). If singularities exist, they are documented and automatically-checked unit tests are provided, so that we can measure quantitatively if algorithms behave as expected. This provides a complete set of results that can be used for comparisons.
- Back to back testing: First we produce "ground truth" output data from any reference implementation: e.g. a Matlab implementation when we are porting to C++ or the original program, when we are implementing a third party algorithm. Then, this ground truth is used to compare against

the output of the ported algorithm (back to back testing). Differences are highlighted and, when possible, corrected. Changes detected by the 'back-to-back' procedure that are not due to implementation errors need to be evaluated using a qualitative criteria to be deemed acceptable or not. When the changes have been validated by other means, the back-to-back tests should be updated to the new value.

- White box unit testing: Use case studies, where results can be theoretically predicted without any computation from reference implementations. Test signals are usually non-realistic but quick and simple to generate and check. Examples include: silences, Dirac deltas, pulses, sounds from synthetic sources, etc. Known cases can be provided as automatically-checked unit tests.

- Accuracy checks and testing: Use a quality measure so that whenever differences appear on the back-to-back testing, we could check whether the results are within a tolerance window of performance. Purpose-made signals and manual annotations (e.g. a sequence of pitches on a test signal, annotated beat, chords, etc) are often used for these quality measurements (e.g. detection accuracy, distortion rate, interference rate, etc). Alternatively, when no automated quality measure can be given, manual guidelines could be provided to allow implementers to validate a changed result (e.g. an expected harmonic structure, a decomposition of the signals into parts that sound in a particular way, etc).

1.5.2 Test data

Data for test cases is on the 'descriptorsData/' folder inside the tarball CLAM-TestData-0.7.0.tar.gz available at `http://www.iua.upf.es/mtg/clam/download.html`

For the testing of semantic descriptors, audio (wav) files are taken as they are, and results are compared against ground-truth annotations (see chapter on evaluation). Purpose-made data for special test cases is placed on each descriptor's specific folder.

For all descriptors there is back-to-back testing for every input data file. Those tests come with CLAM 0.7.0 sources in 'test/UnitTests/DescriptorsTests'.

1.5.3 About cross-platform evaluations

CLAM based algorithms are multiplatform. They run smoothly in both, Linux and Windows, showing mostly negligible precision differences that are checked on a one to one basis. Therefore, successful porting into CLAM implies satisfying the multiplatform requirement of this deliverable.

1.5.4 About usability for WP3 and WP4

The purpose of each semantic descriptor within SIMAC, its clearly stated on the report and on the online documentation (see appendix). Descriptors in this document automatically create semantic information from audio signals (as demonstrated in the WP2 demos). They are used in WP3 and WP5 for research related to the development of the Music Organizer and the Music Recommender.

Applications of these descriptors for the tasks of classification, retrieval, visualisation and browsing are discussed in this report on a case by case basis.

Chapter 2

Describing Rhythm

Rhythm belongs with harmony, melody and timbre as one of the most fundamental aspects of music. Sound by its very nature is temporal, and in its most generic sense, the word *rhythm* is used to refer to all of the temporal aspects of a musical work, whether represented in a score, measured from a performance, or existing only in the perception of the listener. In order to build a computer system capable of intelligently processing music, it is essential to design representation formats and processing algorithms for the rhythmic content of the music.

Computer systems reported in the literature offer different interpretations of the words "automatic rhythm description" as they address diverse applications such as tempo induction, beat tracking, quantisation of performed rhythms, meter induction and characterisation of intentional timing deviations. Although some rhythmic concepts are consensual, no single representation of rhythm has been devised which would be suitable for all applications. Gouyon and Dixon [GD05] present an exhaustive review of the relevant literature and propose a unifying framework for automatic rhythmic description. In this chapter we discuss the issue of rhythm representation and present four different aspects of rhythm description (in increasing order of complexity): onset detection, beat tracking, tempo/based classification and music characterisation according to rhythmic patterns.

2.1 Representing musical rhythm

At the core of automatic rhythmic analysis lies the issue of identifying the start, or onset time, of events in the musical data. A naive approach to describe the rhythm (whether in audio or symbolic data) is to specify an exhaustive and accurate list of onset times, maybe together with some other musical features characterising those events (e.g., durations, pitches and intensities in the MIDI representation). However, such a representation lacks abstraction. There is more to rhythm than the absolute timings of successive musical events. There seems to be agreement on the fact that, in addition, one must also take into account the metrical structure, tempo and timing [Hon01]. However, there is no consensus regarding explicit representations of these three rhythmic concepts.

A first reason is that different rhythmic features are relevant at each step in the musical communication chain, at each step where rhythmic content is produced, transmitted and/or received. As we illustrate in the next sections, metrical structure, tempo and timing take slightly different meanings for composers, performers and listeners. Indeed, even if a goal in the field of music psychology is to seek representational elements, or processes, that would stand as "universal" or "innate" (i.e. functioning at birth, independent of environmental influence) [DB01], a more widespread objective is to determine differences in perception according to listeners' culture, musical background, age or sex [Dra93, Lap00, Gab73, DJB00].

A second reason is that the diverse media used for rhythm transmission suffer a trade-off between the level of abstraction and the comprehensiveness of the representation. Standard Western music notation provides an accepted method for communicating a composition to a performer, but it has little value in representing the interpretation of a work as played in a concert. On the other hand, a MIDI file might be able to represent important aspects of a performance,

but it does not provide the same level of abstraction as the score. At the extreme end, an acoustic signal implicitly contains all rhythmic aspects but provides no abstraction whatsoever. In an application context, the choice of a suitable representation is based on the levels of detail (respectively abstraction) of the various aspects of music which are provided by the representation.

2.1.1 Metrical structure

Western music notation provides an objective regular temporal structure underlying musical event occurrences and organising them into a hierarchical metrical structure. This is independent of the hierarchical phrase structure which may be explicit in the notation or implicit in the composer's, the performer's and/or the listener's conceptualisation of the music.

The Generative Theory of Tonal Music (GTTM, [LJ83a]) formalises this distinction by defining rules for a "musical grammar" which deals separately with grouping structure (phrasing) and metrical structure. While the grouping structure deals with time spans (durations), the metrical structure deals with durationless points in time, the *beats*, which obey the following rules. Beats must be equally spaced. A division according to a specific duration corresponds to a *metrical level*. Several levels coexist, from low levels (small time divisions) to high levels (longer time divisions). There must be a beat of the metrical structure for every note in a musical sequence. A beat at a high level must also be a beat at each lower level. At any metrical level, a beat which is also a beat at the next higher level is called a downbeat, and other beats are called upbeats. Beats obey a discrete time grid, with time intervals all being multiples of a common duration, the smallest metrical level.

Music psychology research asserts that humans perceive at least part of the objective temporal structure. Drake and Bertrand [DB01] advocate a universal "predisposition toward simple duration ratio", and claim that "we tend to hear a time interval as twice as long or short as previous intervals." The Dynamic Attending Theory [DPB00, JB89] proposes that humans spontaneously focus on a "referent level" of periodicity, and they can later switch to other levels to track events occurring at different time spans (for instance, longer-span harmony changes, or a particular shorter-span fast motive). However, metrical structure perception is strongly dependent on musical training [DPB00].

2.1.2 Tempo

Given a metrical structure, *tempo* is defined as the rate of the beats at a given metrical level, for example the quarter note level in the score. There is usually a *preferred* or *primary metrical level*, which corresponds to the rate at which most people would tap or clap in time with the music, and this is commonly used to define the tempo, expressed either as a number of beats per minute, or as the time interval between beats (the *inter-beat interval*). In many cases the primary metrical level corresponds to the denominator of the time signature, and the next one or two higher levels are specified by the numerator of the time signature.

However, the perception of tempo exhibits a degree of variability. It is not always correct to assume that the denominator of the time signature corresponds to the "foot-tapping" rate, nor to the actual "physical tempo" that would be an

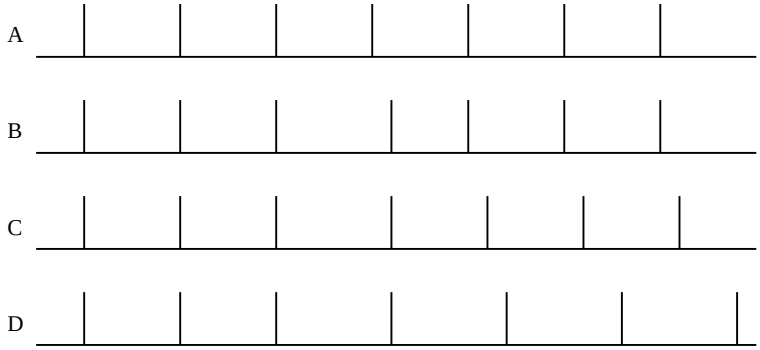


Figure 2.1: Four time-lines, marked with onsets, illustrating the difference between tempo and timing changes: (A) an isochronous pulse; (B) a local timing change; (C) a global timing change; and (D) a tempo change.

inherent property of audio flows [DGP99]. Differences in human perception of tempo depend on age, musical training, musical preferences and general listening context (e.g. tempo of a previously heard sequence, listener's activity, instant of the day) [Dra93, DPB00, Lap00, DJB00]. Differences in tempo perception are nevertheless far from random; they most often correspond to a focus on a different metrical level, e.g. differences of half or twice the inter-beat interval (when hearing duple meter music) or one-third or three times the inter-beat interval (when hearing triple or compound meter music).

2.1.3 Timing

Although it is supposed to model the listener's intuitions, a major weakness of the GTTM is that it does not deal with the departures from strict metrical timing which are apparent in almost all styles of music. Thus it is only really suitable for representing the timing structures of musical scores, or as an abstract representation of a performance, where the expressive timing is not represented.

There are conceptually two types of non-metrical timing, which come under the headings *tempo* and *timing* respectively. These are illustrated in Figure 2.1, which shows a strictly metrical (isochronous) pulse (A), followed by three variations on this pulse. There are two types of timing changes: in the first case (B), just one beat in the pulse is displaced, whereas in the second case (C), all beats from a particular time onwards are displaced, as when a pause occurs in the music. In both of these cases, the change is in the timing; there is a discontinuity in the pulse, but the rate of the pulse on both sides of the discontinuity is the same. In this sense we can associate timing changes with short term changes in the pulse. On the other hand, a tempo change is a change in the rate of the pulse (D), which is a long term change in the pulse.

It is important to note that at the time of the first change (the 4th beat in the pulse), it is impossible to distinguish cases (B), (C) and (D). This makes causal analysis impossible (i.e. algorithms which do not use information about future

events in analysing present events, as, for example, any real-time algorithm), since with no knowledge of the future, a single "out of time" beat could be due to either a tempo or timing change [CDGW01].

One of the greatest difficulties in analysing performance data is that the two dimensions of tempo and timing are projected onto the single dimension of time. Mathematically, it is possible to represent any tempo change as a series of timing changes and any timing change as a series of tempo changes, but these descriptions are somewhat counterintuitive [Hon01]. The parsimony of the representation is an important factor in its psychological plausibility [Tan93].

In order to represent changing tempi, various approaches can be used. If tempo is considered as an instantaneous value, it can be calculated as the inter-beat interval measured between each pair of successive beats. A more perceptually plausible approach is to take an average tempo measured over a longer period of time. A measure of central tendency of tempo over a complete musical excerpt is called the *basic tempo* [Rep94], which is the implied tempo around which the expressive tempo varies. The end result of any of these approaches is a value of tempo as a function of time, which is called a tempo curve. Often, timing is also modelled by the tempo curve representation, an approach which is sharply criticised [DH91, Hon01] for failing to separate the dimensions of tempo and timing. This criticism is well supported by examples where transformations applied to a tempo curve representation do not preserve musically important features.

Among others, Bilmes [Bil93] and Baggi [Bag91] propose to represent timing deviations as systematic event shifts occurring within the span of the fastest pulse, while keeping a constant execution speed. They found evidence of the suitability of such a representation in analysing respectively Latin percussion music and jazz music. Friberg and Sundström [FS02, FS99] propose to focus on the swing. The term originates in Jazz music, and is often characterised by the long-short pattern of performing consecutive eighth-notes. The *swing ratio* refers to a mathematical expression: the duration of the first eighth-note divided by that of the second.

Music psychology research presents evidence that listeners perceive performers' intentional timing deviations. Clarke [Cla87] shows that "categorical perception" differentiates expressive timing from rhythmic structure: a small number of categories are used to characterise the *continuously* variable temporal transformation of the *discrete* (integer ratio) structure. Further, timing and structure are tightly linked. Repp [Rep92] confirms listeners' sensitivity to timing deviations, but, most importantly, also shows that this sensitivity is a variable of the position in the metrical structure. Complementary to this finding, there is strong evidence that performers do not produce timing deviations at arbitrary points in time [Pal97]. They rather deviate from pure mechanical performance in specific ways. The metrical structure provides "anchor points" for timing deviations, and "every aspect of musical structure contributes to the specification of an expressive profile for a piece" (Clarke [Cla99], p.492). Expressive timing is also systematic; the timing in repeated performances can be very stable over a period of years (Clynes and Walker [CW82], pp.181-187).

The chief goal in automatic rhythm description is the parsing of acoustic events that occur in time into the more abstract notions of metrical structure, tempo and timing, as illustrated in Figure 2.2, where the goal is to derive a

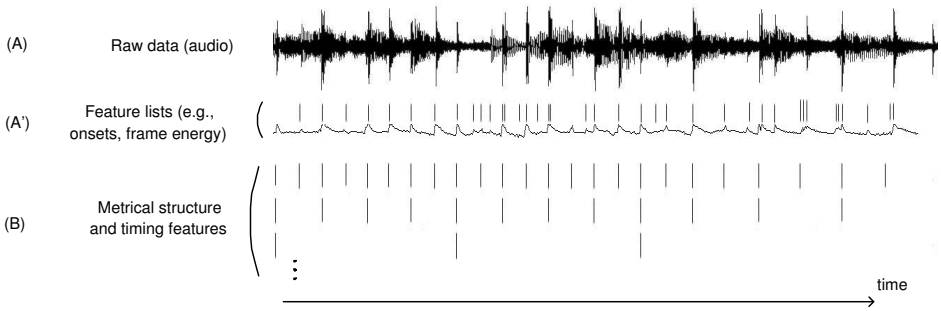


Figure 2.2: Example of an audio signal, examples of feature lists, corresponding metrical structure and timing features, showing a gradually decreasing tempo.

representation like (B) from (A) or (A'). This is made very difficult by the inherent ambiguity of rhythm [GD05], as discussed in the previous section.

In the following we present four approaches, developed under SIMAC, for the automatic description of rhythm from raw audio data. We start from the low-level of onset detection and progressively increase the complexity of the description until we finalise by describing an approach to music characterisation using automatically estimated rhythmic patterns.

2.2 Onset detection: temporal segmentation of musical events

Temporal segmentation of audio into note events is useful for a range of audio analysis, editing and synthesis applications. Examples may include automatic transcription, content analysis and non-linear time-scale modification and pitch-shifting. The segmentation task is especially difficult for complex mixtures including both percussive and non-percussive onsets. In musical signals, let us assume notes to be events defined by the temporal concatenation of an attack transient: a short and unpredictable segment characterized by fast changes in the intensity, pitch or timbre of the sound [MR97]; followed by the steady-state of the signal, where it is stationary, thus easily predictable. An onset can be defined as the instant when the attack transient begins, thus marking the beginning of the note. Typically, note onset detection schemes use energy-based approaches, often involving frequency weighting [Mas96]. In recent years, this has been extended to include sub-band schemes such as [RJ01, Kla99].

Here we describe work presented in [BDDS04], where we depart from the basic theory of energy-based onset detection, extending our analysis to include phase information, then combining both methods in a complex domain approach that improves experimental results while providing a more robust theoretical framework.

2.2.1 Energy-Based Onset Detection

Usually, the introduction of a new note leads to an increase in the energy of the signal. In the case of strong percussive note attacks, such as drums, this increase

in energy will be very sharp. For this reason, energy has proved to be a useful, straightforward, and efficient metric by which to detect percussive transients, and therefore certain types of note onset. The local energy of a frame of the signal $s(m)$ is defined as:

$$E(m) = \sum_{n=(m-1)h}^{mh} |s(n)|^2 \tag{2.1}$$

where h is the hop size, m the hop number and n is the summation variable. Taking the first difference of $E(m)$ produces a detection function from which peaks may be picked to find onset locations. This is one of the simplest approaches to note onset detection. The idea can be extended to consider frames of an FFT.

Let us consider a time-domain signal $s(mh)$, whose STFT is given by:

$$S_k(m) = \sum_{n=-\infty}^{\infty} s(n)w(mh-n)e^{-j2\pi nk/N} \tag{2.2}$$

where $k = 0, 1, \ldots, N-1$ is the frequency bin index and $w(n)$ is a finite-length sliding window. It follows that the magnitude difference between consecutive FFT frames is then:

$$\delta S = \sum_{k=1}^{N} |S_k(m)| - |S_k(m-1)| \tag{2.3}$$

This measure, known as the spectral difference, can be used to build an effective onset detection function (an implementation based on this can be found in [DSD02]). Energy-based algorithms are fast and easy to implement, however their effectiveness decreases when dealing with non-percussive signals and when transient energy overlaps in complex mixtures. Energy bursts related to transient information are more noticeable at higher frequencies as the "tonal" energy is usually concentrated at lower frequencies, masking the effect of these variations on the signal content.

2.2.2 Phase-based onset detection

The Short-Time Fourier Transform of the signal, $S_k(m)$, can also be defined in terms of a group of sinusoidal oscillators with time-varying amplitudes $|S_k(m)|$ and phases $\varphi_k(m)$ (with unwrapped phases denoted as $\tilde{\varphi}_k(m)$). During the steady-state part of the signal these oscillators will tend to have constant frequencies. Therefore, the difference between two consecutive unwrapped phase values must remain constant between frames. For the k^{th} oscillator[1] that is:

$$\Delta\tilde{\varphi}(m) = \tilde{\varphi}_e(m) - \tilde{\varphi}(m-1) = \tilde{\varphi}(m-1) - \tilde{\varphi}(m-2) \tag{2.4}$$

where $\tilde{\varphi}_e(m)$ is the estimated unwrapped phase for the current frame. Ideally, a target phase can be defined as:

$$\tilde{\varphi}_t(m) = \tilde{\varphi}(m-1) + \Omega_k h \tag{2.5}$$

[1]For clarity, the sub-index k is not included in the equations of 2.2.2.

where Ω_k is the frequency of the k^{th} sinusoid. However, only synthesized sounds in artificially-controlled conditions behave like this. For real sounds we expect a deviation phase to be added to the target in order to generate the estimated unwrapped phase. This deviation can be calculated as:

$$\tilde{\varphi}_d(m) = \text{princarg}[\tilde{\varphi}(m) - \tilde{\varphi}_t(m)] \tag{2.6}$$

where the function princarg maps the phase to the $[-\pi, \pi]$ range. By calculating the estimated unwrapped phase as:

$$\tilde{\varphi}_e(m) = \tilde{\varphi}_t(m) + \tilde{\varphi}_d(m) \tag{2.7}$$

and substituting Equations 2.5, 2.6 and 2.7 in Equation 2.4, we can obtain values for the difference of unwrapped phase values. Collecting all terms into the right hand side of Eq. 2.4, we can obtain an expression for the phase deviation between target and the real phase values in a given frame:

$$d_\varphi = \text{princarg}[\tilde{\varphi}(m) - 2\tilde{\varphi}(m-1) + \tilde{\varphi}(m-2)] \tag{2.8}$$

d_φ will tend to zero if the current phase value is close to the estimated value and will deviate significantly from zero otherwise. The latter is the case for most oscillators during attack transients.

Let us extend this analysis to the distribution of phase deviations for all oscillators within one analysis frame. Let us call $f_m(d_{\varphi,k})$ the probability density function of our data set on a particular frame m. During the steady-state part of the signal most values are expected to be concentrated around zero, creating a sharp distribution. On the other hand, during attack transients, the corresponding distributions will be flat and wide. In [BS03b], these observations are quantified by calculating the inter-quartile range and the kurtosis coefficient of the distribution. Here we propose measuring the frame-by-frame spread of the distribution as:

$$\eta_p(m) = \text{mean}(f_m(|d_{\varphi,k}|)) \tag{2.9}$$

Measuring $\eta_p(m)$ is a fast and reliable approach to generating a detection function. Phase-based onset detection offers an alternative to common energy-based methods, overcoming detection constraints for soft onsets. However, the method is susceptible to phase distortion and to the variations introduced by the phase of noisy components (usually related to low-energy values which are therefore ignored).

2.2.3 Detection of onsets in the complex domain

There are a number of reasons that justify combining phase and energy information for onset detection: while energy-based approaches favor strong percussive onsets, phase-based approaches emphasize soft, "tonal" onsets; the two methods are more reliable at opposite ends of the frequency axis; the information they gather behaves in a similar statistical manner. A first attempt to combine these approaches was presented in [DBDS03]. Then, measures of spread for both distributions were simply multiplied, compensating for instabilities in either approach and producing sharper peaks for detected onsets. However, this analysis does not imply a fully combined approach where energy and phase information

is simultaneously analyzed. This can only be achieved in the complex domain as will be explained in the following.

For locally steady state regions in audio signals, we can assume that frequency and amplitude values remain approximately constant. In the previous sections it has been shown that by inspecting changes in either frequency and amplitude, onset transients can be located. However, by predicting values in the complex domain, the effect of both variables can be considered. Let us assume that, in its polar form, the target value for the k^{th} bin of the STFT is given by:

$$\hat{S}_k(m) = \hat{R}_k(m)e^{j\hat{\phi}_k(m)} \tag{2.10}$$

where the target amplitude $\hat{R}_k(m)$ corresponds to the magnitude of the previous frame $|S_k(m-1)|$, and the target phase $\hat{\phi}_k(m)$ can be calculated as the sum of the previous phase and the phase difference between preceding frames:

$$\hat{\phi}_k(m) = \text{princarg}[2\tilde{\varphi}_k(m-1) - \tilde{\varphi}_k(m-2)] \tag{2.11}$$

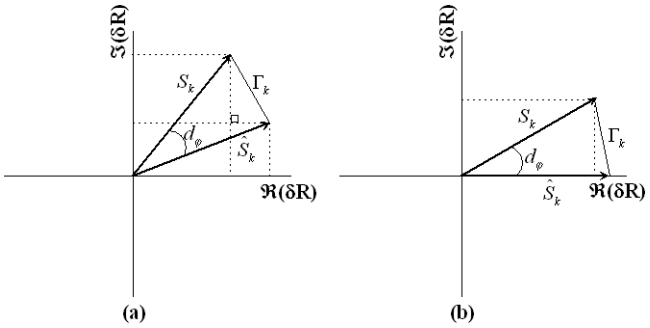


Figure 2.3: Phasor diagram in the complex domain showing the phase deviation between target and current vector, and the Euclidean distance between them: normal diagram (a) and rotated diagram (b).

We may then consider the measured value in the complex domain from the STFT:

$$S_k(m) = R_k(m)e^{\phi_k(m)} \tag{2.12}$$

where R_k and ϕ_k are the magnitude and phase of the current STFT frame. By measuring the Euclidean distance between target and current vectors in the complex space, as shown in Figure 2.3 (a), we can then quantify the stationarity for the k^{th} bin as:

$$\begin{aligned} \Gamma_k(m) \quad = \quad & \Big\{\big[\Re\big(\hat{S}_k(m)\big) - \Re\big(S_k(m)\big)\big]^2 + \\ & \big[\Im\big(\hat{S}_k(m)\big) - \Im\big(S_k(m)\big)\big]^2\Big\}^{\frac{1}{2}} \end{aligned} \tag{2.13}$$

Summing these stationarity measures across all k, we can construct a frame-by-frame detection function as:

$$\eta(m) = \sum_{k=1}^{K} \Gamma_k(m) \tag{2.14}$$

Equation 2.14 can be simplified by mapping $\hat{S}_k(m)$ onto the real axis (forcing $\hat{\phi}_k(m) = 0$), such that:

$$\hat{S}_k(m) = \hat{R}_k(m) = R_k(m-1) \tag{2.15}$$

This implies rotating the phasors, as shown in Fig. 2.3 (b), so that $S_k(m)$ can be represented using the phase deviation (Eq. 2.8):

$$S_k(m) = R_k(m)e^{jd_{\varphi k}(m)} \tag{2.16}$$

Let us consider the difference between this complex domain prediction approach and the basic amplitude difference measure in equation 2.3. This can be re-written as:

$$\delta S_k(m) = \hat{R}_k(m) - R_k(m) \tag{2.17}$$

With the mapping onto the real axis of $\hat{S}_k(m)$, Eq. 2.13 becomes:

$$\begin{aligned}
\Gamma_k(m) &= \big\{\big[\hat{R}_k(m) - \Re\big(S_k(m)\big)\big]^2 + \Im\big(S_k(m)\big)^2\big\}^{\frac{1}{2}} \\
&= \big\{\big[\hat{R}_k(m) - R_k(m)cos\big(d_{\varphi k}(m)\big)\big]^2 + \\
&\quad \big[R_k(m)sin\big(d_{\varphi k}(m)\big)\big]^2\big\}^{\frac{1}{2}} \\
&= \big\{\hat{R}_k^2(m) - 2\hat{R}_k(m)R_k(m)cos\big(d_{\varphi k}(m)\big) + \\
&\quad R_k^2(m)sin^2\big(d_{\varphi k}(m)\big) + R_k^2(m)cos^2\big(d_{\varphi k}(m)\big)\big\}^{\frac{1}{2}} \\
&= \big\{\hat{R}_k(m)^2 + R_k(m)^2 - \\
&\quad 2\hat{R}_k(m)R_k(m)cos\big(d_{\varphi k}(m)\big)\big\}^{\frac{1}{2}}
\end{aligned} \tag{2.18}$$

For the case of $d_{\varphi k}(m) = 0$:

$$\begin{aligned}
\Gamma_k &= \big\{\hat{R}_k^2(m) + R_k^2(m) - 2\hat{R}_k R_k\big\}^{\frac{1}{2}} \\
&= \hat{R}_k(m) - R_k(m)
\end{aligned} \tag{2.19}$$

Therefore $\Gamma_k(m)$ is only equal to $\delta S_k(m)$ when $d_{\varphi k}(m)$ is equal to zero, e.g. when the phase prediction is "good". In that case, only the energy difference is being taken into account. In the case of $d_{\varphi k}(m) \neq 0$, the phase deviation from the prediction is taken into account. $\eta(m)$ constitutes an adequate detection function showing sharp peaks at points of low stationarity. Figure 2.4 depicts the detection function for a section of a guitar signal. The figure also gives examples of phase and amplitude used individually. The complex domain approach is clearly less noisy, therefore simplifying the task of peak-picking and allowing a more robust detection.

2.2.4 Peak-picking

To enhance the selection of peaks in the detection functions, the median filter is used to obtain an adaptive threshold curve $\delta_t(m)$. This is calculated as the weighted median of an H-length section of the detection function around the corresponding frame, such that:

$$\delta_t(m) = \delta + \lambda \,\text{median}\, \eta(k_m), k_m \in [m - \frac{H}{2}, m + \frac{H}{2}] \tag{2.20}$$

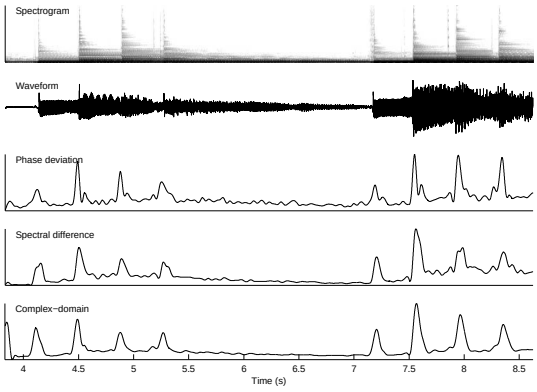


Figure 2.4: Spectrogram of a music signal (upper) and onset detection functions using phase-based (upper-middle), energy-based (lower-middle) and the proposed complex-domain (bottom) approaches.

δ and λ are constant values, however while the latter is only a scaling factor, variations of the former largely affect the good and false detections ratio. [Kau02] demonstrated the effectiveness of the median filter for the thresholding of peaks in detection functions generated from music.

2.2.5 Results

Experimental results compare the three presented approaches to onset detection: the spread of the distributions of spectral differences and phase deviations, and the complex-domain approach. The spread measure is used as it was found to be the most efficient and effective as shown in [DBDS03]. The experiments were performed on a 1065-onsets database of hand-labelled music segments.

For all methods, Figure 2.5 displays the percentage of good detections versus the percentage of false positives for $\lambda = 1$ and different δ values (scaled by -10^2). Better performance shifts the curve up and leftwards. The complex-domain approach outperforms the other two methods. Its curve's optimal point[2] ($\{90.2, 5.0\}_{\delta=5.65}$) is above those of the spectral difference ($\{83.0, 4.1\}_{\delta=8.21}$) and the phase deviation ($\{81.8, 5.6\}_{\delta=4.27}$). It reaches the highest values of correct detections and is only outperformed by the energy method at the bottom of the plot. For low detection rates the spectral difference outperforms the phase deviation, that presents high rates of false positives.

Table 2.1 shows results according to onset types: pitched non-percussive (e.g. bowed strings), pitched percussive (e.g. piano), non-pitched percussive (e.g. drums) and complex mixtures (e.g. pop music). Results are obtained by using the optimal δ value (scaled by -10^2) for each method in each case (from the corresponding performance curve). Results support that, for most cases, combining energy and phase information outperforms either approach alone. The only exceptions being firstly, the PNP case where the phase method

[2] the point representing the fewest errors for a given δ by being closest to 100% correct detections and 0% false positives

	PNP			PP		
METHOD	δ	**OK**	**FP**	δ	**OK**	**FP**
Spectral difference	4.58	87.1	8.6	9.62	94.9	1.6
Phase-based	0.06	95.7	4.3	6.62	95.5	0.3
Complex-domain	4.58	92.5	8.8	5.95	98.8	2.6
	NPP			CMIX		
METHOD	δ	**OK**	**FP**	δ	**OK**	**FP**
Spectral difference	1.50	81.6	5.5	7.02	81.2	10.7
Phase-based	1.16	80.7	5.5	2.33	80.1	24.7
Complex-domain	0.34	94.3	5.6	5.79	84.1	9.3

Table 2.1: Onset detection results for: pitched non-percussive (PNP), pitched percussive(PP), non-pitched percussive (NPP) sounds and complex mixtures (CMIX). Columns show the corresponding δ values (scaled by -10^2), and the percentages of correct detections (OK) and false positives (FP) per method.

METHOD	**COMPUTATIONAL COST**
Spectral difference	1.90
Phase-based	1.80
Complex-domain	2.38

Table 2.2: Computational cost per frame for each method normalized to the load of the FFT algorithm.

performs best, as quantifying only tonal changes is best for music with soft onsets; and secondly, the PP case where the phase-based algorithm returns less false positives than the complex-domain (at a higher total error rate). Spectral difference and phase-based methods are prone to under and over-detections (due mostly to amplitude modulations and overlapping for the first, and to phase distortion and frequency modulations for the second) especially when dealing with complex mixtures.

Figure 2.6 shows how the complex-domain approach also provides better time localization for onsets. It shows percentages of good detections for different comparison windows (between target and detected events) on a database of acoustic recordings of MIDI-generated piano music (thus minimizing the error introduced by hand-labelling). The optimal δ values for pitched-percussive music were used. It supports quantitatively the argument made (2.2.3) regarding the sharpness of the different detection functions. These results demonstrate that the theoretical robustness of the complex-domain approach implies also a practical advantage over the other methods.

Finally, Table 2.2 analyzes computational expense of the algorithms. Computational cost is calculated using the quantity of FLOPS (floating point operations) per frame (averaged across different executions), where the values have been normalized such that the computational cost of performing the FFT alone is set to 1. For all three methods, the increase in computation over the basic FFT algorithm is small enough to allow real-time implementation. The most expensive algorithm, the complex-domain method, is only 1.32 times the cost

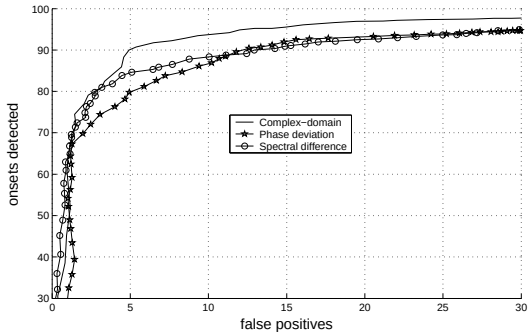


Figure 2.5: Percentage of onset detections versus percentage of false positives for different values of δ (the peak-picking threshold): using complex-domain, phase deviation and spectral difference detection methods.

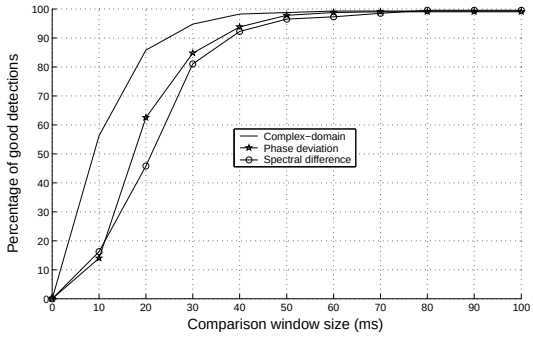


Figure 2.6: Percentage of good detections for different lengths of the comparison window (ms): using complex-domain, phase-based and spectral difference detection methods. δ values correspond to the PP case in Table 2.1.

of the phase-based algorithm, the fastest of them all.

2.2.6 Discussion

Energy-based onset detection schemes perform well for pitched and non-pitched music with significant percussive content. On the other hand, phase-based onset detection approaches provide better results for strongly pitched signals (even for "softer" onsets), while being less robust to distortions in the frequency content and to noise. In the complex domain, both phase and amplitude information work together, offering a generally more robust onset detection scheme. Therefore, the presented theory for the complex domain approach to onset detection is not just an evolution of the spectral difference and the phase-based approaches, but a more general framework, in which the others are particular cases.

From the practical point of view, it is straightforward to implement while remaining computationally cheap. Additionally it proves effective for a large range of audio signals, as experimental results corroborate.

Beyond the task of detecting the onset times, it is important to note that, the onset detection function, as shown in figure 2.2, is a valuable signal representation as it encodes information pertinent to the location and strength of musical events in the signal. This information defines rhythm at its lowest level.

Although, as pointed in the previous section, it is too naive to rely on the sole use of the detection function to describe rhythm, it can prove useful as an intermediate representation towards the generation of higher levels of semantic description. In the following sections we will exemplify this, by using the complex-domain onset detection function as the basis of a beat tracking system and by including energy-based detection functions as part of the feature set used to characterise rhythmic patterns.

2.3 Context-dependent beat tracking

Having illustrated a technique for temporal segmentation of musical events, we then extend our analysis to the identification of beats in musical signals, using the onset detection function as our primary input.
The principal aim of a beat tracking system is to replicate the human ability of tapping in time to music. In the simplest context, a successful algorithm should be able to extract two parameters from an audio signal: a measure of the rate at which beats occur (which we will call the *beat period*) and identifying when they occur (i.e. finding their phase, or *beat alignment*). This seemingly simple problem, at which humans, even without any musical knowledge, seem particularly adept, does not translate so trivially into a computational environment, especially in situations where the tempo of the input varies.
Under certain conditions however, the problem of beat tracking becomes less complicated. This appears directly related to the number of assumptions made about the characteristics of the input signal. Goto's approach [Got01] is an excellent example. By constraining all input signals to be in 4/4 time and assuming approximately constant tempo falling within the range of 61 and 120 bpm (beats per minute) his system is able to perform very robust beat tracking in real-time. However when attempting the design of a more general beat tracking system, performance naturally decreases. The problem which poses the greatest difficulty is that of tempo variation occurring as a result of natural variation as well intentional timing changes, known as *expressive timing*.
A number of broad approaches to beat tracking have been proposed, including those which employ probabilistic formulations to the problem [CKDH00a, Hai04] as well as those more grounded in signal processing techniques such as comb filter analysis [Sch98a, Kla03b]. The approach we present [DP05a] borrows from both of these fields. We use comb filtering techniques in conjunction with a state space switching model to perform accurate and efficient beat tracking using a General and Context Dependent Model.

2.3.1 General Model

In this section we first describe the general approach taken towards beat tracking as presented in [DP04]. Beat locations are derived from a two stage process, which begins by identifying the beat period, followed by a separate beat alignment stage using the current period estimate. The analysis is frame based, using

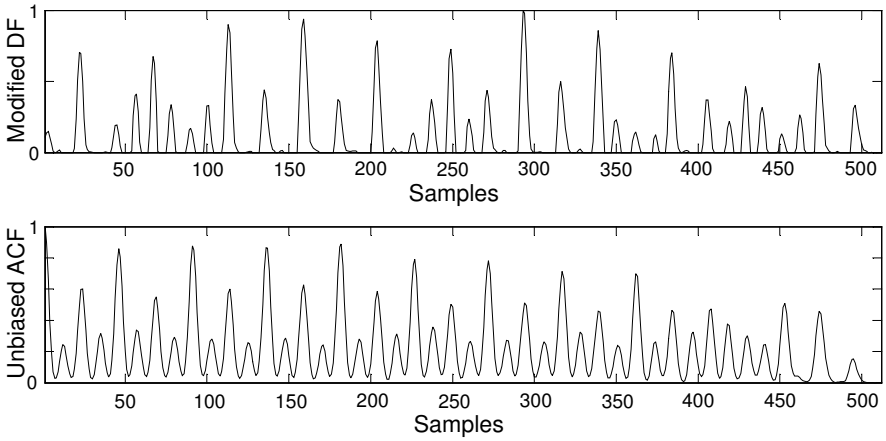


Figure 2.7: Modified Detection Function (DF) (top) and Unbiased Autocorrelation Function (ACF) (bottom)

a window of 512 onset detection function samples (whose resolution is 11.6ms) with a step increment of 128 samples, giving an overlap of 75%. The ability to successfully extract beats from audio relies on having a long enough window to accurately identify the tempo, while maintaining the flexibility necessary to follow timing variations.

Onset Emphasis

The onset detection function (DF) can be considered a continuous representation of onset *emphasis* wherein the local maxima, or peaks of the signal represent the onset locations - a property we shall exploit to identify beat locations.
As seen in the previous section, a number of approaches to generating the detection function exist, where different properties of the audio signal are emphasised. We select the complex domain approach [BDDS04] as it is sensitive both to *tonal* and *percussive* events within musical audio signals.
The detection function $df[n]$ is low pass filtered and subjected to an adaptive median threshold to yield a Modified DF (top plot, Fig 2.7) as the input to the system.

Beat Period Estimation

The beat period, τ, is found by identifying the most salient lag l from an *unbiased* autocorrelation function (ACF), $\hat{r}_{df}[l]$, of the current detection function frame (of length N)

$$\hat{r}_{df}[l] = ((\sum_{n=0}^{N-1} df[n]df[n-l])(|(l-N)|)) \tag{2.21}$$

where lag (in DF samples) can be converted to tempo (in bpm) using the relation: $tempo = 60/(l * 0.0116)$, and 11.6ms is the resolution of the DF.

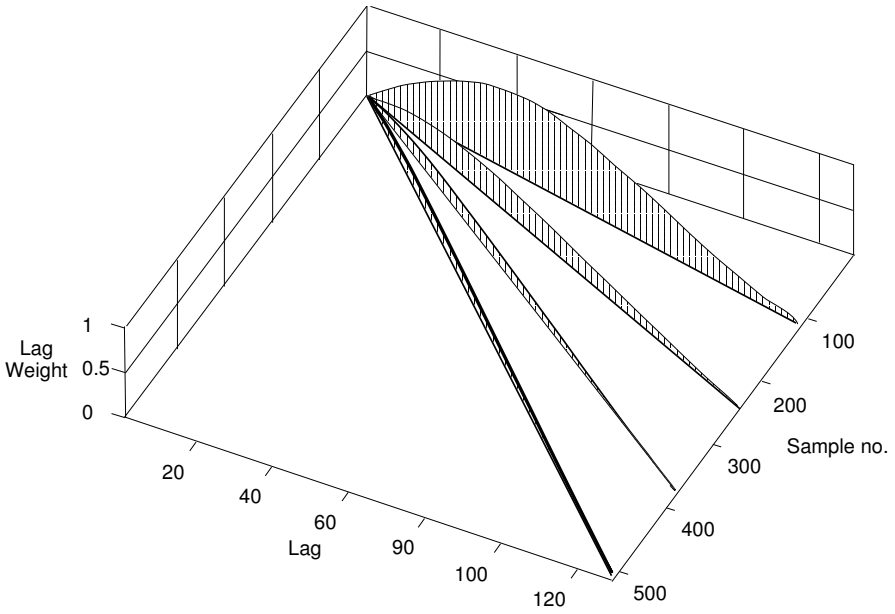


Figure 2.8: 3-D view of Lag Weighted Comb Filterbank, M

The ACF is passed through a *shift-invariant* comb filterbank, M, covering a lag range of 1 to 128 samples (Fig 2.8). The elements of M grow wider at each harmonic but are normalised such that each harmonic has equal influence over the location of the beat period. This prevents any metrical bias. Each row of M is scaled by $R_w[l]$, a lag weighting function derived from the Rayleigh Distribution Function, with parameter $b = 43$ (corresponding to the *preferred* tempo of 120 bpm [vNM99])

$$R_w[l] = \frac{l}{b^2} e^{\frac{-l^2}{2b^2}} \tag{2.22}$$

This weighting, which is similar to the resonance model for beat period [vNM99] acts as a *band pass filter* to encourage the beat period to be extracted in the approximate tempo octave range of 80 to 160 bpm. The inclusion of R_w effectively transforms the goal of the General Model into that of *preferred beat tracking* as signals with tempi outside the effective range of the weighting function will observed at a metrical level within this range.
An accurate value for τ (which overcomes the inverse tempo to lag relationship) is found by finding the column of M which best matches the ACF, i.e. $\arg\max_l(\hat{r}_{df} \times M)$. This column is then used as a *mask* on the ACF with

$$\tau = \frac{1}{P} \sum_{p=1}^{P} \frac{h_p}{p} \tag{2.23}$$

where P is the number of harmonics in M and h_p is the index of the local maxima of the ACF for the p^{th} harmonic.

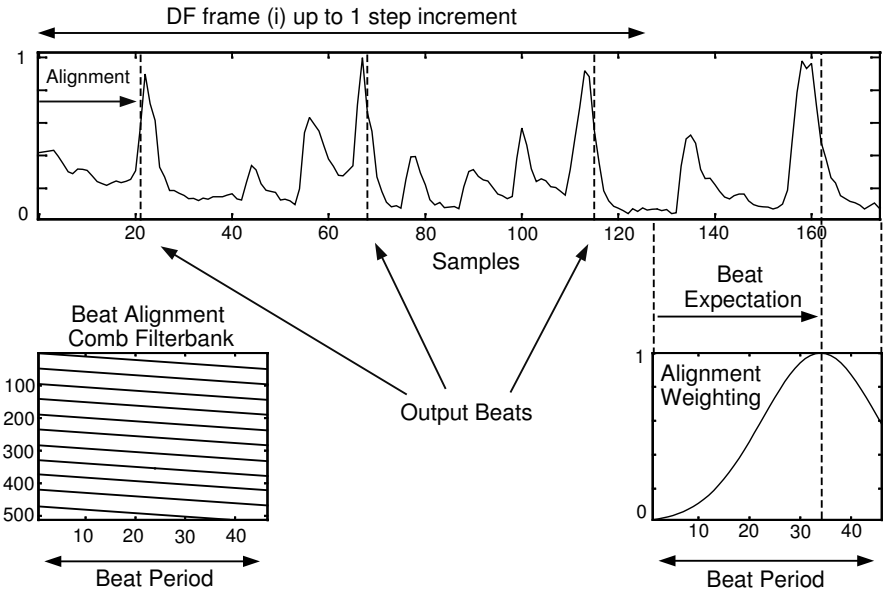


Figure 2.9: Beat Placement and Alignment Weighting

Beat Alignment and Placement

Having found the beat period for the current analysis frame, this value can be used to find the beat alignment, ϕ, defined as the offset from the start of the current frame, t_i, to where the *first* beat should occur.
The method for identifying ϕ is similar to the way in which τ was found, as a comb filter matrix is again used. However instead of the passing the ACF (which by definition is *zero-phase*) through the comb filter, the DF frame is used instead. The properties of the *alignment* matrix comb filter, A (shown in the bottom left of Fig 2.9) differ from those of M as the elements, which occur at integer multiples of τ allow a search through all possible shifts of τ in the range t_i to $t_i + \tau$ with ϕ found using:

$$\phi = \arg\max_n(df[n] \times A) \tag{2.24}$$

Beats within the current DF frame can now be placed up to *one* step increment beyond t_i using the following relationship

$$\gamma_m = (t_i + \phi) + (m - 1)\tau \tag{2.25}$$

where γ_m is m^{th} beat in the current frame.

2.3.2 Context Dependent Model

When tested in [DP04] the General Model demonstrated promising results which confirmed the overall approach as appropriate for the problem of beat tracking. However the system's primary failure was in producing a consistent output. The rate at which beats occurred was found to frequently (and unpredictably) switch

between metrical levels as well as switching between on-beats to off-beats locations. Given the importance of consistency both in terms of possible applications for a beat tracking system, and the emphasis given in evaluation procedures for the longest continuous segment [GM97a, Kla03b, Hai04] we chose to refine the General Model to make better use of the available rhythmic information. We therefore propose a Context Dependent Model.

Beat and Measure Period

Within the Context Model we replace the Rayleigh weighting applied to M by a *beat period dependent* Gaussian lag weighting, $G_w[l]$

$$G_w[l] = e^{\frac{-(l-\tau)^2}{2\sigma^2}} \tag{2.26}$$

where the mean corresponds to an average beat period for the input (which could be derived from a number of individual measurements using the General Model or equally from a notated value, if available). The variance, $\sigma^2 = 3.9$ was empirically derived by analysis of the distribution of *tempo normalised* inter beat intervals from the hand labelled beat database used in evaluation. It has an important property in that it allows for some expressive timing variation, but is narrow enough to give a consistent output, even when the input ACF displays very little noticeable periodicity.
In addition to incorporating G_w a simple test is performed to infer the measure level periodicity within the ACF. Currently this involves the classification between *duple* and *triple*, i.e. music that is in either 4/4 or 3/4 time. The decision is made by evaluating the following condition related to energy in the ACF: $(2\tau + 4\tau > 3\tau + 6\tau)$. If true, we assume duple time, else triple time. This then enables the structure of M to be altered such that it consistent with the meter classification. A metrical bias can now be imposed into M by setting the appropriate number of harmonics (either 3 or 4) and scaling them to give most emphasis to detecting measure level periodicity from which the beat periodicity can then be inferred.

Beat Expectation

The problems observed in phase-switching were a direct result of the *memoryless* state of the General Model. By making independent judgements about beat alignment at every frame, the alignment value typically locked to the strongest peak in the first period of the DF, which could have resulted from an on-beat, off-beat or an accented event unrelated to the beat structure. We aim to resolve this problem, by the generation of an adaptive Gaussian alignment weighting to apply to each row of A, the beat alignment matrix, thus invoking the concept of *beat expectation*. The procedure for generating the alignment weighting is shown in Fig 2.9. In the General Model, beats were placed only up to one step increment beyond the start of the current analysis frame. However, assuming that the observed beat period will not vary too greatly (a valid assumption given the narrow Gaussian weighting, $G_w[l]$), a beat prediction can be made by identifying the location of the *next* possible beat, i.e. $\gamma_{last} + \tau$ which acts as the mean of the Gaussian alignment weighting. The variance of this weighting was derived to prevent phase-switching, and $\sigma^2 = \tau/4$ was found to be ideal.

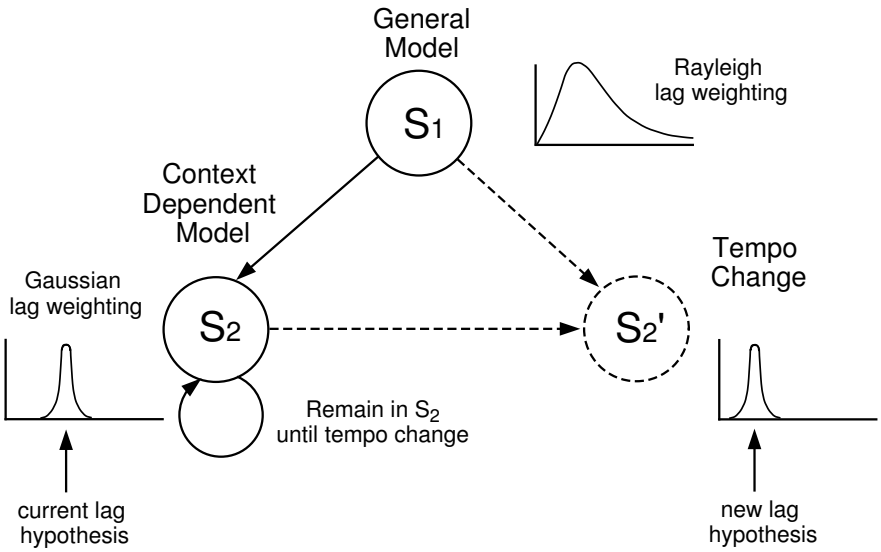


Figure 2.10: State Switching Model

Algorithm Used	CML Cont.	CML Total	AML Cont.	AML Total
Davies - GM	26.4	50.4	29.6	59.8
Davies - TSM	53.6	60.8	66.1	81.8
Hainsworth [Hai04]	45.1	52.3	65.5	80.4
Scheirer [Sch98a]	23.8	38.9	29.8	48.5
Klapuri [Kla03b]	55.9	61.4	71.2	80.9

Table 2.3: Results comparing different beat tracking algorithms under four conditions: CML - Correct Metrical Level; AML - Allowed Metrical Level; Cont. - Longest Continuous segment, Total - Total number of correct beats

2.3.3 State Switching

Having presented both the General and Context Dependent Models we now seek to combine the two, and describe the conditions under which state transitions occur. In the initial instance, there has been no observation of beat period from which to form a Context Model. Therefore beat analysis begins in state S_1 (see Fig 2.10) using the flat alignment weighting. The Context Model is generated (and hence the transition from S_1 to S_2 occurs) as a result of observing three consecutive *consistent* beat period values such that:

$$\mathrm{abs}(2\tau_\mathrm{r}(\mathrm{i}) - \tau_\mathrm{r}(\mathrm{i}-1) - \tau_\mathrm{r}(\mathrm{i}-2)) < \eta \tag{2.27}$$

where $\tau_r(i)$ is the beat period using the Rayleigh weighting for frame i, and $\eta = \sigma^2$ from eq. (2.26).
Beat placement now uses period measurements taken from the Context Model in conjunction with the beat expectation alignment weighting (sec. 2.3.2). We

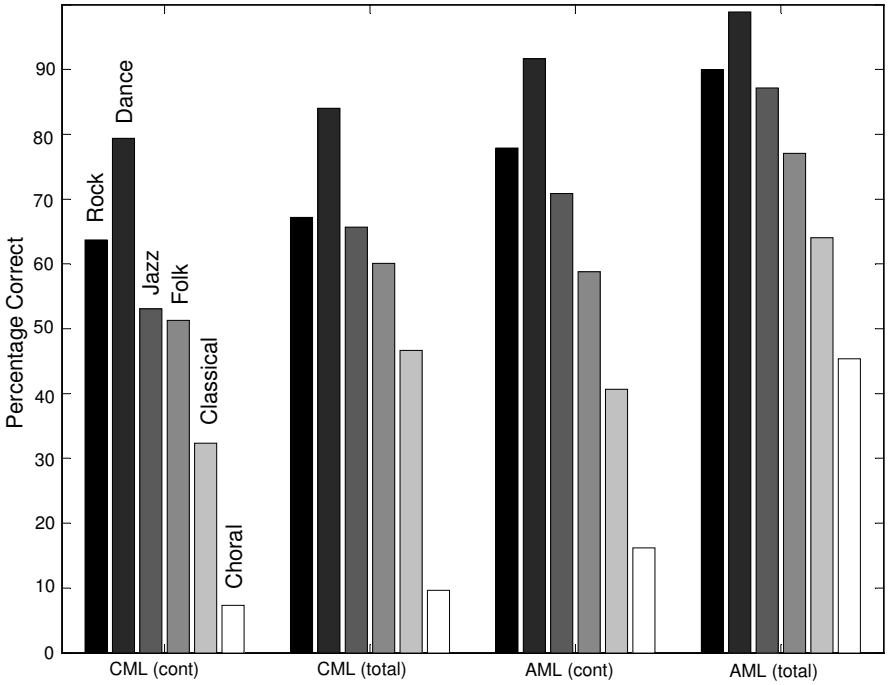


Figure 2.11: Comparison of results across musical genre

remain in state S_2 until the observation of a new *consistent* lag hypothesis, defined by:

$$\mathrm{abs}(\tau_{\mathrm{r}}(\mathrm{i}) - \tau_{\mathrm{g}}(\mathrm{i})) > \eta \tag{2.28}$$

where the consistency condition is as in eq. (2.27) and $\tau_g(i)$ is beat period from the Context Model. Once a new beat period hypothesis has been selected, (i.e. the transition from S_2 to S_2') state S_2 is no longer valid and is removed from the system.

2.3.4 Results

We evaluated our algorithm by quantitatively comparing beat locations generated by our algorithm to a database where ground truth beat locations had been hand-labelled by a musician. The annotation was split into two parts: first claps were recorded in time to the audio (and their temporal indices extracted); each annotated track was then subjectively analysed and any mis-placed beats were manually corrected. The database contained 222 audio tracks, each around a minute in length, covering the following range of musical genres: rock/pop (68), dance (40), jazz (40), folk (22), classical (30) and choral (22) [Hai04]. Annotation is a time-consuming and difficult process which is at the core of the design and evaluation of all rhythmic descriptors. Thus, in SIMAC we have proposed a semi-automatic tool for the annotation of rhythmic content in audio signals. Its details are explained in Appendix A.
The primary performance metric used for evaluation is defined as the ratio of the *longest continuous correctly tracked* segment to the length of the input signal,

and has been used in several beat tracking studies [GM97a, Kla03b, Hai04]. Beats are considered to have met this criterion if they occur within ±20% of the annotated beat onset time and the previous and subsequent beats are placed within ±25% of their respective annotated values. In addition to this rather strict method, a second criterion is defined as the *total number of correct beats*, where the continuity requirement is relaxed. In order to allow for the metrical ambiguity that occurs when humans tap along to music - that in some cases beats are tapped at twice or half the correct level, or on the off-beat instead of the on-beat, results are re-calculated to allow for this behaviour. Therefore to summarise, we use a total of 4 criteria: i) the longest continuous segment, or *Cont.* at the correct metrical level, *CML*; ii) the *total* number of correct beats, at *CML*; iii) the longest continuous segment at the allowed metrical levels *AML* and iv) the *total* number of correct beats at *AML*. Table 2.3 shows the performance of our *Two State Model* in comparison with the General Model, as well as the implementations of: Hainsworth [Hai04], Scheirer [Sch98a] and Klapuri [Kla03b]. A breakdown of the performance of our system is shown in Fig 2.11 for each genre within the test set.
As can be seen from the results in Table 2.3, a significant improvement has been achieved by the inclusion of the Context Model, such that the Two State Model clearly outperforms the General Model. In terms of a direct comparison with the other published algorithms, the General Model achieves equivalent results to the Scheirer implementation [Sch98a], with the Two State Model showing better performance than that of Hainsworth's approach [Hai04], but marginally poorer than that of Klapuri [Kla03b], in all categories except the total number of correct beats at *AML*.
The results shown in Fig 2.11 demonstrate an expected pattern across musical genres with the best performance in the rock/pop and dance categories. The most prominent failure of the algorithm is in addressing choral music, where there is often very little, if any coherent beat structure. The algorithm also relies on the tempo of the input to remain constant within each analysis frame - an assumption which does not hold in cases where there is considerable expressive timing. Our approach to meter analysis is not particularly robust, as it can only address music which is in 3/4 or 4/4 time. The extraction of *downbeat* indices (the first beat of each bar) should also improve performance by adding some higher level structure to the system.

2.3.5 Discussion

We have applied a two state switching model to the problem of beat tracking for musical audio signals. The inclusion of a context dependent model has been shown to significantly improve upon the results achieved in our previous work [DP04]. In particular with respect to maintaining a consistent metrical level and preventing phase switching between off-beats and on-beats.
Areas for further investigation include expanding our beat tracking model to address multiple metrical levels, for sub-beat analysis and grouping beats in bars. We also expect that further improvements can be made by incorporating an adaptive input selection stage able to match an appropriate detection function to the properties of the input signal (e.g. energy-based for percussive music - as in Section 2.2.1 -, phase-based for non-percussive music - as described in Section 2.2.2).

2.4 A word on music classification using high-level rhythmic features

As previously discussed, most music can be described in terms of dimensions such as melody, harmony, rhythm, instrumentation and form. These high-level features characterise music and at least partially determine its genre, but they are difficult to compute automatically from audio. As a result, most audio-related music information retrieval research has focussed on extracting low-level features and then using machine learning to perform tasks such as classification. This approach has met with some success, but it is limited by two main factors: (1) the low level of representation may conceal many of the truly relevant aspects of the music; and (2) the discarding of too much information by the feature extraction process may remove information which is needed for the accurate functioning of the system.

In SIMAC's Work Packages 2 and 3, we address some aspect of these limitations by extracting high-level descriptors directly from audio. Examples have been presented on the previous two sections, where detection functions, onsets and beats were automatically generated from the signal. These representations, can also be used for the generation of increasingly complex descriptions of the rhythm of musical pieces. An example, in Section 2.3.2, is the use of the onset detection function for beat tracking. Another example, not explicitly discussed, is the estimation of the overall tempo of a musical segment as defined by the temporal distances between tracked beats.

In the following two sections, we attempt to explore ever higher levels of semantic analysis, related to the issues of music similarity and classification which are at the core of WP3. First we explain an approach to dance music classification using tempo as the main musical descriptor, and then we expand this technique by introducing the notion of rhythmic patterns for the characterisation of music.

2.5 Using Tempo for Dance music classification

Tempo is a musical attribute of prime importance. Moreover, recent research [GDPW04a] demonstrates its relevance in the task of classifying different styles of dance music: focusing *solely* on the *correct* tempo (i.e. measured ***manually***) 8 classes of Standard and Latin ballroom dance music can be classified, by means of diverse classification techniques, with over 80% accuracy (total of 698 instances, classes are listed in Table 2.4, baseline is 15.9%).

In this section we propose to further exploit the high relevance of the tempo in designing a classifier that focuses first on this feature and then uses complementary features to make decisions in possibly ambiguous situations. Performance is compared to that reported in [GDPW04a] (a 1-NN classifier with 15 MFCC-like descriptors, and *no* tempo information, yielded **79%** correct classification).

The work reported here is based on the assumption that, given a musical genre, the tempo of any instance is among a very limited set of possible tempi. For instance, the tempo of a Cha Cha is usually between 116 and 128 BPM. Table 2.4 gives tempo ranges for the 8 dance styles used in this experiment. This assumption may be arguable, yet it seems to make sense for ballroom

dance music as, on the one hand, common musical knowledge (e.g. instructional books) suggests such boundaries, and on the other hand, [Moe02] shows on a large amount of data (more than 90000 instances) that different dance music styles ("trance, afro-american, house and fast") show clearly different tempo distributions, centered around different "typical" tempi.

Cha Cha	$116 - 128$
Jive	$160 - 180$
Quickstep	$198 - 210$
Rumba	$90 - 110$
Samba	$96 - 104$
Tango	$120 - 140$
Viennese Waltz	$170 - 190$
Slow Waltz	$78 - 98$

Table 2.4: Dance music tempo ranges, in BPM.

We define a Gaussian tempo probability function for each class. The Gaussian standard deviations are defined so that the probabilities at the limits specified in Table 2.4 are half the value of the corresponding probability maximum. Put together, these probabilities may overlap in certain tempo regions (e.g. Samba and Rumba, see dashed-blue and solid-black lines around 100 BPM in Figure 2.12). Hence, given an unknown instance, a simple classification process could be:

1. Compute its tempo T
2. Retrieve the n classes that overlap significantly at T
3. Use a classifier tailored to these n classes

Assuming different tempo distributions, it is reasonable to consider that much less than 8 classes do overlap *significantly* at any tempo. The classifier design is consequently easier than the eight-class learning task considering all examples.

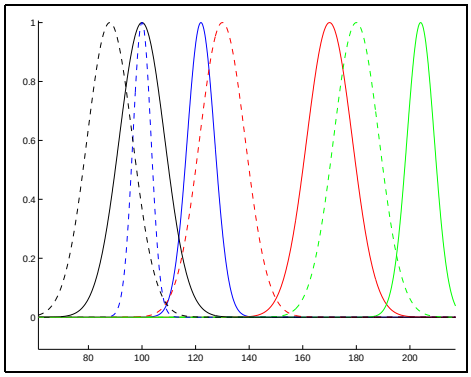


Figure 2.12: Tempo probability functions and overlaps of 8 dance music styles. X-axis in BPM.

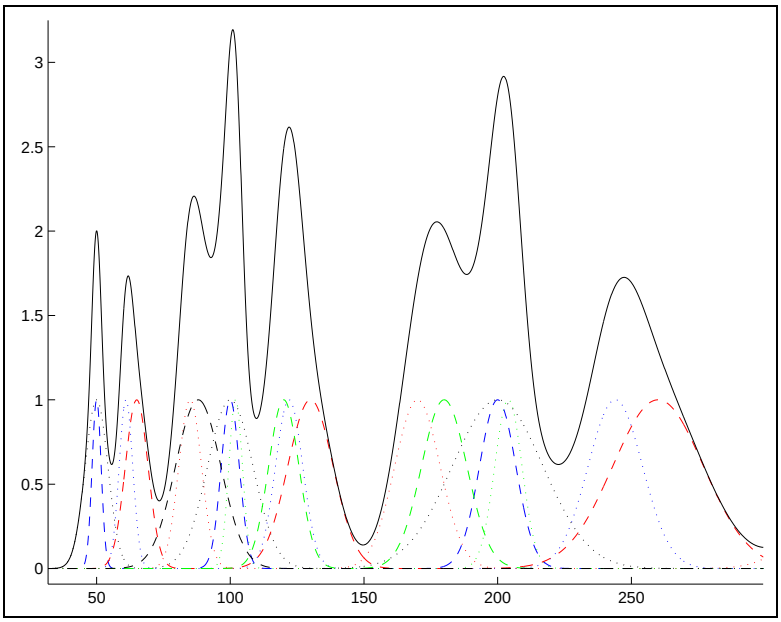


Figure 2.13: Adapted tempo probability functions of 8 dance music styles. X-axis in BPM. Solid black line is the sum of all probability functions and represents overall class overlaps.

However, there is a consensus in the tempo-tracking literature on the fact that state-of-the-art tempo induction algorithms typically make errors of metrical levels (they output e.g. half the correct tempo). This is true both for the tempo derived from the beat tracker in Section 4.2, as for the implementation used here, Dixon's BeatRoot [Dix01a]. This algorithm may output the correct tempo, twice or half of its value (in the case of excerpts with a duple meter) or two thirds of its value (in the case of excerpts with a triple meter). For further tempo induction evaluation details, on the database used here, we refer to [GDPW04a] and to appendix A. We adapted the tempo probability functions accordingly in concatenating several Gaussians whose means are correct tempi and relevant multiples (see Figure 2.13).

Observing the probability functions in Figure 2.13, one can see that each tempo value corresponds usually to two different potential classes, at the exception of three specific tempo regions in which three classes overlap. These are 95 to 105 BPM and 193 to 209 BPM for Quickstep, Rumba and Samba, and 117 to 127 BPM for Cha Cha, Tango and Viennese Waltz. Therefore, we propose to build 30 different classifiers:

- $\sum_{n=1}^{8-1} n = 28$ two-class classifiers, $\{K_1 \dots K_{28}\}$, each expert in a specific pairwise classification task.
- 2 three-class classifiers, K_{29} and K_{30}, each expert in a three-class specific task

When presented with unknown instances, the knowledge available to the system is this set of 30 expert classifiers and the tempo probability functions

for all possible classes. Therefore, the overall classification process is finally:

1: Compute tempo T of the instance to classify
2: Find the classifier K_i whose tempo range includes T
3: Perform classification with K_i

2.5.1 Descriptors

The descriptors used here are detailed in [GDPW04a]. Tempo-related descriptors have been removed (the correct tempo and 3 different implementations). The 70 remaining features describe low-level characteristics of 2 different periodicity representations, the "Periodicity Histogram" and the "Inter-Onset Interval Histogram", e.g. flatness, skewness, MFCC-like, etc., see [GDPW04a].

For each classifier, we evaluated descriptor relevances on an individual basis (i.e. Ranker search method associated to ReliefF attribute evaluator), and selected the 10 most relevant features. All experiments have been conducted with Weka (*http://www.cs.waikato.ac.nz/ml/weka*).

2.5.2 Classification results

For classification, we use Support Vector Machines as it is commonly suggested for two-class problems. All percentages result from 10-fold cross-validation procedures. Systematic evaluation of different classification methods is left to future work.

The majority of the 28 pairwise classifier accuracies are above 90%, all using 10 descriptors. The worst classification is that between Slow Waltz and Viennese Waltz (81.8% accuracy, baseline 63%). The best is that between Quickstep and Viennese Waltz (100% accuracy, baseline 55.7%). Regarding the three-class classifiers, also using 10 descriptors, K_{29} (Quickstep vs. Rumba vs. Samba) has 84.1% accuracy (baseline 36.6%) and K_{30} (Cha Cha vs. Tango vs. Viennese Waltz) 91.9% accuracy (baseline 42.3%). To measure the overall accuracy of the 30 classifiers, let us compute a weighted average of their individual accuracy. The weights are proportional to the number of times a classifier is actually required (given the tempo estimations of the 698 excerpts). This yields **89.4%** accuracy.

Let us now evaluate the whole classification process. Recall that, as the process involves 2 steps, it suffers from tempo estimation errors in addition to misclassifications. In 24.3% of the cases (i.e. 170 excerpts) the tempo estimation step assigns excerpts to pairwise (or three-class) classifiers that do *not* account for its true class. There is no way to recover from these errors, whatever the subsequent classification, the excerpt will be assigned to an incorrect class.

The overall accuracy of the system is therefore the multiplication of both step accuracies, i.e. $0.894 \times 0.757 = \mathbf{67.6\%}$. One might wonder whether considering metrical level errors in the design of the tempo probabilities actually results in any improvement. As reported in [GDPW04a], the tempo induction algorithm has around 50% accuracy (considering multiples as errors). The resulting overall accuracy is therefore around $0.894 \times 0.5 = 44.7\%$. The improvement is over 20%.

However, we noted that tempo induction is especially bad for Slow Waltz excerpts, yielding around 75% to be assigned to wrong classifiers. This is because onset detection, in the tempo induction algorithm, is designed for percussive

onsets, which are often lacking from waltzes. An improvement could be obtained by using phase-based approaches such as the one in Section 2.2.2. If we remove the Slow Waltz excerpts from the database, 587 remain, and the number of excerpts that are assigned to irrelevant classifiers falls to 13.9%. The overall accuracy rises now to **76.5%**.

2.5.3 Discussion

In this section, we have investigated the classification of music from 8 different dance styles, using an approach that puts a special emphasis on the tempo estimation. The proposed classification process entails two steps: tempo computation and use of expert (pairwise or three-class) classifiers in specific tempo regions. We show that it is possible to design very accurate expert classifiers. However, in this framework, if the tempo estimation fails, the classification fails. We show that a substantial classification improvement can be obtained when considering tempo multiples.

The accuracy on a database of 698 excerpts from 8 classes is 67.6%. Restricting tests to the 7 classes (587 excerpts) on which tempo estimation is reasonably reliable, the accuracy increases to 76.5%. This is slightly worse than results reported in [GDPW04a] with a different method (an eight-class classifier yielded 79% accuracy with 15 *tempo-independent* descriptors).

In conclusion, reducing the problem from an eight-class learning task to several two- or three-class learning tasks is only pertinent when having an *extremely* reliable tempo estimation algorithm. To illustrate this, consider the accuracy of our method (whose expert classifier can still be improved) when using the correct tempi (measured manually): 82.1%. All this clearly indicates that while tempo is an important attribute, it is not sufficient, especially when relying on the automatic estimation of semantic descriptors directly from the raw audio. Therefore, in the following section we present an approach that improves results by using structures derived from beats (rhythmic patterns), rather than tempo alone.

2.6 Rhythmic patterns for music characterisation

2.6.1 Introduction

In this section we present a new method of characterising music by typical bar-length rhythmic patterns which are automatically extracted from the audio signal, and demonstrate the usefulness of this representation by its application in a genre classification task. Recent work has shown the importance of tempo and periodicity features for genre recognition, and we extend this research by employing the extracted temporal patterns as features. Standard classification algorithms are utilised to discriminate 8 classes of Standard and Latin ballroom dance music (698 pieces). Although pattern extraction is error-prone, and patterns are not always unique to a genre, classification by rhythmic pattern alone achieves up to 50% correctness (baseline 16%), and by combining with other features, a classification rate of 96% is obtained [DGW04a]. These classification rates represent a significant improvement over the method in Section 2.5, which

is evaluated on the same data set, and higher rates than have been published on other data sets [TC02, MB03, DPW03].

It is hypothesised that rhythmic patterns are not randomly distributed amongst musical genres, but rather they are indicative of a genre or small set of possible genres. Therefore, if patterns can be extracted successfully, we can test this hypothesis by examining the usefulness of the patterns as features for genre classification. As dance music is characterised by repetitive rhythmic patterns, it is expected that the extraction of prominent rhythmic patterns would be particularly useful for classification. However, rhythmic patterns are not necessarily unique to particular dance styles, and it is known that there is a certain amount of overlap between styles. In this work, rather than assuming a fixed dictionary of patterns, we use an automatic extraction technique which finds the most salient pattern for each piece. Thus the techniques used are generalisable to other musical genres.

First the bar length patterns in the amplitude envelope are found and clustered using the k-means algorithm with a Euclidean distance metric. The centre of the most significant cluster is used to represent the piece, and a feature vector consisting of this rhythmic pattern and various derived features is used for classification on a music database of the first 30 seconds of 698 pieces of Standard and Latin ballroom dance music.

Although we focus solely on the rhythmic aspects of music, we show that for genre classification of dance music, a very high level of accuracy is obtainable. The results show an improvement over previous methods which used periodicity or inter-onset interval histograms and features derived from these. Other possible applications of automatically extracted rhythmic patterns are query and retrieval of music, playlist generation, music visualisation, synchronisation with lights and multimedia performances.

In the following section we outline the background and related work, and then in the subsequent sections describe the pattern extraction algorithm and genre classification experiments, concluding with a discussion of the results and future work.

2.6.2 Related Work

Audio feature extraction was first addressed in the context of speech recognition, and was later applied to classification tasks in order to separate speech from non-speech signals such as music and environmental sounds [Sau96, WBKW96]. More recently, several authors have addressed classification tasks specific to music, such as instrument recognition [HPD03a] and detection of segments of music that contain singing [BE01]. Others have focussed on determining similarity judgements for content-based retrieval [Foo97], for the organisation and navigation of large music collections [Pam01, PDW04] and for computation of high level semantic descriptors [Sch00].

Automatic musical genre classification has a shorter history. Tzanetakis et al. [TEC01, TC02] used three sets of features representing the timbral texture, rhythmic content and pitch content of musical pieces, and trained statistical pattern recognition classifiers to achieve a 61% classification rate for ten musical genres. McKinney and Breebart [MB03] examined the use of various low level feature sets and obtained 74% classification on 7 musical genres. Dixon et al [DPW03] compared two methods of periodicity detection, and developed a

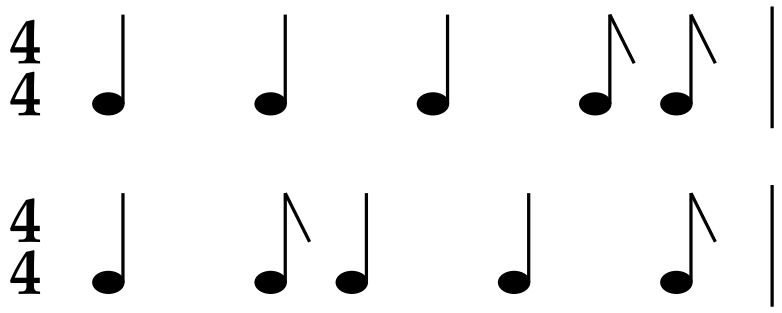


Figure 2.14: The importance of temporal sequence: these two different rhythm patterns have the same distribution of inter-onset intervals but are typical of two different genres, Cha Cha (above) and Rumba (below).

simple rule based system to classify 17 styles of Standard and Latin ballroom dance music based only on the distribution of periodicities, with an accuracy of 37%.

In the abovementioned work, limited rhythmic information is encoded in the beat histogram [TC02], modulation energy [MB03] or periodicity distribution [DPW03]. Each of these representations provides information about the relative frequency of various time intervals between events, but discards the information about their temporal sequence. For example, consider the two rhythmic patterns in Figure 2.14. Both patterns have the same distribution of inter-onset intervals (three quarter notes and two eighth notes), but the patterns are perceptually very different. The upper pattern, which is typical of a Cha Cha rhythm, would not be described as syncopated, whereas the lower pattern, more likely to be found in a Rumba piece, is somewhat syncopated.

Rhythmic patterns were used by Chen and Chen [CC98] for song retrieval using symbolic queries and a database of symbolic music, in which approximate string matching provided the similarity measure. The patterns used were not general patterns which summarise a piece or a genre, but specific patterns which did not need to occur more than once in the piece.

The only work we are aware of in which rhythmic patterns were automatically extracted from audio data is by Paulus and Klapuri [PK02], who extracted bar-length patterns represented as vectors of loudness, spectral centroid and MFCCs, and then used dynamic time warping to measure similarity. Their work did not include genre classification, although they did indicate that the similarity of drum patterns was higher within genre than between genres.

Other relevant research that involves the extraction of rhythmic content from a musical performance is beat tracking [GM95a, GM99a, Sch98b, CKDH00b, Dix01b], that is, finding the times of the beats (at various metrical levels). If we assume that rhythmic patterns exist within a particular metrical unit, e.g. within bars, then finding the boundaries of these metrical units becomes a prerequisite to pattern finding. The main difficulty in beat tracking is not in finding the periodicities, but their phase. That is, the length of a pattern can be estimated much more reliably than its starting point. We use an interactive

Genre	Pieces	Metre	Tempo (nominal)	Tempo (actual)
Cha Cha	111	4	128	92–137
Jive	60	4	176	124–182
Quickstep	82	4	200–208	189–216
Rumba	98	4	104	73–145
Samba	86	4	200	138–247
Tango	86	4	128–132	112–135
Viennese Waltz	65	3	174–180	168–186
Waltz	110	3	84–90	78–106

Table 2.5: Statistics of the data used in the experiments. Tempo is given in BPM, where a beat corresponds to a quarter note (except for Samba and some Viennese Waltzes which have an eighth note beat). Nominal tempo values are according to the overviews at the `www.ballroomdancers.com` web site.

beat tracking system [Dix01c] in order to annotate the first bar of each piece.

2.6.3 Pattern Extraction

Data

Two major difficulties for developing music information retrieval systems are the lack of reliably labelled data sets, and the fuzziness of class boundaries of the attributes. Ballroom dance music has the advantage of providing a set of genres for which there is a high level of agreement among listeners concerning the genre. We collected 698 samples of Standard and Latin ballroom dance music (`http://www.ballroomdancers.com`), each consisting of approximately the first 30 seconds of a piece. The music covers the following eight classes: Cha Cha, Jive, Quickstep, Rumba, Samba, Tango, Viennese Waltz and (Slow) Waltz. The distribution of pieces over the classes, the nominal tempo of each class, and the actual tempo ranges of the excerpts are shown in Table 2.5.

Audio Processing

The samples were uncompressed from Real Audio format to a standard PCM format at the same sampling rate as the original file (either 44100, 16000 or 11025 Hz, always mono). The amplitude envelope was extracted from the signal using an RMS filter. The frame rate was set so that a bar would contain a fixed number b of samples at any tempo (where the tempo is already known, as described below).

If $x(n)$ is the input signal with sampling rate r and bar length l seconds, then its amplitude envelope is calculated with a sampling rate of b samples per bar using a hop size h given by:

$$h = \frac{rl}{b} \tag{2.29}$$

The amplitude envelope $y(n)$ is given by:

$$y(n) = \sqrt{\frac{\sum_{i=nh}^{(n+k)h-1} x(i)^2}{kh}} \tag{2.30}$$

where k is the overlap factor. The bar lengths l ranged from 0.97 to 3.30 sec. Best results were obtained with $b = 72$ and $k = 2$, although values of b from 48 to 144 gave similar results.

Two alternative representations for $y(n)$ were also tried, by taking respectively the square and the absolute value of the signal $x(n)$, passing it through an eighth order Chebyshev Type I lowpass filter, and decimating to a sampling rate of b samples per bar. The choice of representation had only a small influence on results.

Bar Finding

Much research has been conducted on beat tracking, that is, finding the times of musical beats in audio files [GM95a, GM99a, Sch98b, CKDH00b, Dix01b]. Although beat tracking is not a solved problem, the extraction of periodicities is reliable, with the remaining difficulties being the mapping of periodicities to metrical levels (e.g. estimating which periodicity corresponds to the rate of quarter notes), and choosing the correct phase for a metrical level (e.g. estimating which quarter note beats correspond to the first beat of each bar).

Since the focus of this work was not to perform beat or measure finding, we used values for the first bar generated by BeatRoot [3] [Dix01c] and corrected manually. This also allowed us to skip irregular (i.e. tempo-less) introductions, which are difficult to detect automatically in short (30 sec) excerpts.

Once the first bar was known, the process of finding subsequent bars was performed automatically, by searching within ±5% of the end of the previous bar for a starting point which has maximum correlation with the sum of previous bars. That is, for each bar i, a correction factor $\delta(i)$ was calculated which determined the offset of the beginning of the following bar $m(i+1)$ from its expected position $(m(i)+b)$. If $d = \lfloor \frac{b}{20} \rfloor$ and $m(i)$ is the index of the beginning of the ith bar, where $m(1)$ is given by BeatRoot, then:

$$m(i+1) = m(i) + b + \delta(i) \tag{2.31}$$

where

$$\delta(i) = \arg\max_{k=-d}^{d} \sum_{j=0}^{b-1} y(m(i)+b+k+j) * z(i,j) \tag{2.32}$$

and

$$z(i,j) = \sum_{k=1}^{i} y(m(k)+j) \tag{2.33}$$

Extracting Rhythmic Patterns

Once the bar positions were determined, bar length rhythmic patterns were then extracted, consisting of the amplitude envelope of the signal between the start

[3] BeatRoot is GPL software available at `www.oefai.at/~simon/beatroot`

and end points of the bar. The ith pattern v_i is a vector of length b:

$$v_i = \langle y(m(i)), y(m(i)+1), \ldots, y(m(i)+b-1) \rangle \tag{2.34}$$

In order to remove outliers, k-means clustering (with $k = 4$) was used to find clusters of similar bars, and the largest cluster was taken as defining the most prominent rhythmic pattern for each piece.

If C_j is the cluster containing the most bars, then the characteristic rhythmic pattern $p(n)$ of a piece is given by:

$$p(n) = \frac{1}{|C_j|} \sum_{k \in C_j} y(m(k)+n) \tag{2.35}$$

Furthermore, we can define the distance $D(i,j)$ between two rhythmic patterns $p_i(n)$ and $p_j(n)$ by the Euclidean metric:

$$D(i,j) = \sqrt{\sum_{k=1}^{b} \left(p_i(k) - p_j(k)\right)^2} \tag{2.36}$$

For example, Figure 2.15 shows the pattern vectors of all 15 bars of one Cha Cha excerpt, where the colours indicate the clusters to which the bars belong, and the thick black curve shows the centre of the largest cluster, that is, the extracted pattern $p(n)$. The perceptual onset of a sound occurs slightly before its energy peak [VR81], so it is valid to interpret peaks occurring immediately after a metrical boundary as representing an onset at that metrical position. For example, the extracted pattern in Figure 2.15 has a peak at each eighth note, clearly implying a quadruple metre, and if the five highest peaks are taken, the resulting pattern corresponds to the upper rhythmic pattern in Figure 2.14.

Viewing the representative patterns for each song provides some feedback as to the success of the pattern extraction algorithm. If the measure finding algorithm fails, the chance of finding a coherent pattern is reduced, although the clustering algorithm might be able to separate the pre-error bars from the post-error bars. The remainder of this section gives examples of extracted rhythmic patterns which have features typical of the genres they represent.

Figure 2.16 shows another Cha Cha piece which has a rhythmic pattern very similar to the one shown in Figure 2.15. By thresholding below the level of the highest 5 peaks, we again obtain the prototypical Cha Cha rhythmic pattern shown in the upper part of Figure 2.14.

Music for Jive and Quickstep is usually characterised by swing eighth notes. That is, each quarter note is broken into an unequal pair of "eighth" notes, where the first is longer than the second. The ratio of the lengths of the two notes is known as the *swing ratio*, which is typically around 2:1. Figure 2.17 shows an extracted pattern where a swing ratio around 2:1 is clearly visible.

One of the characteristics of Rumba is the use of syncopation in the percussion instruments. Accents on the 4th and 6th eighth notes are typical, and this is seen in many of the extracted patterns, such as in Figure 2.18. This pattern is similar (but not identical) to the rhythm shown in the lower part of Figure 2.14.

Finally, the two Waltz patterns in Figure 2.19 clearly show a triple metre, distinguishing these pieces from the other patterns which have a quadruple or duple meter. However, we also note that these two patterns are quite dissimilar,

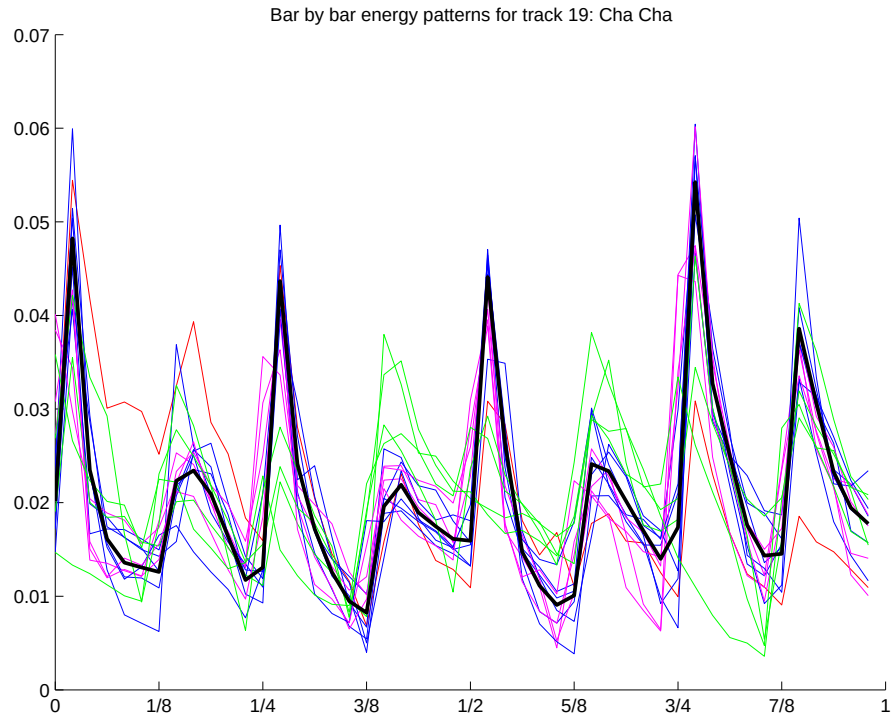


Figure 2.15: The amplitude envelope of the fifteen bars of excerpt 19 are shown, with the colours representing the four clusters. The thick black line is the centre of the largest cluster, that is, the rhythmic pattern which is extracted for this excerpt. This pattern is somewhat typical of the Cha Cha. The labels on the x-axis (showing musical units) were added for illustrative purposes, and were not known to the system.

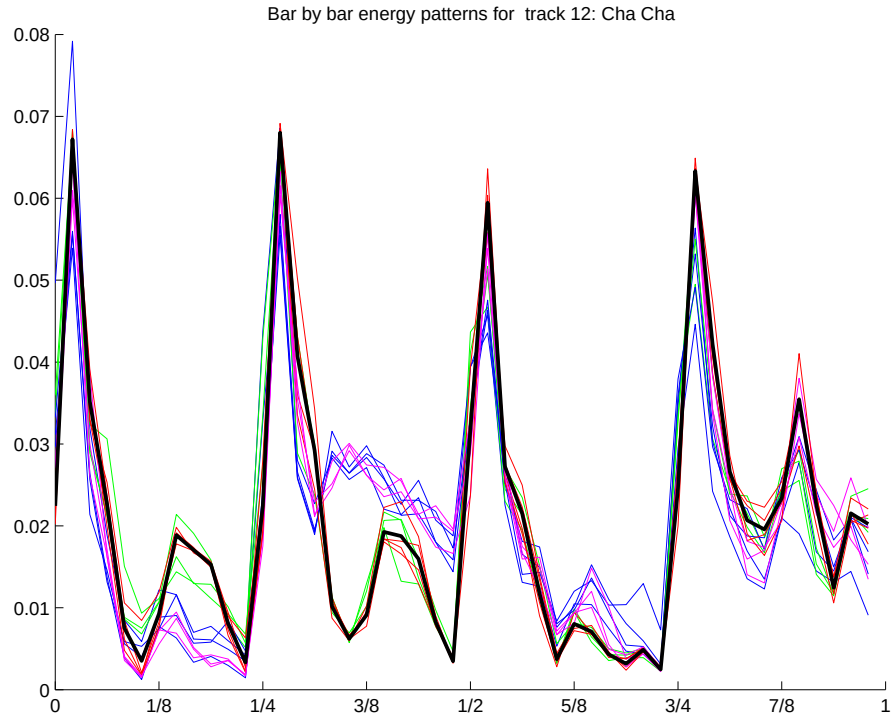


Figure 2.16: Another Cha Cha piece, which has a pattern very similar to the piece in Figure 2.15.

in that the upper one has peaks for each quarter note, whereas the lower pattern has peaks for each eighth note. It is also noticeable in Figure 2.19 that there is much greater variability between the bars of each piece. The lack of prominent percussion instruments makes the amplitude peaks less pronounced, making bar finding and pattern extraction less reliable. As a result, a number of the Waltz patterns failed to show any regularity at all.

2.6.4 Genre Classification Experiments

The relevance of the discovered patterns was evaluated in several genre (dance style) classification experiments. Various supervised learning algorithms and data representations (see below) were compared empirically. Classification accuracy was estimated via a standard 10-fold cross-validation procedure: in each experiment, the training examples were randomly split into 10 disjoint subsets (folds), 9 of these folds were combined into a training set from which a classifier was induced, and the classifier was then tested on the remaining tenth fold; this was repeated 10 times, with each fold serving as test set exactly once.

Classification was performed with the software Weka[4] [WF99], using the following classification algorithms. The simplest method used was the k-Nearest Neighbours (k-NN) algorithm. For $k = 1$ this amounts to assigning each test set instance to the class of the nearest element in the training set. For $k > 1$,

[4] `www.cs.waikato.ac.nz/ml/weka`

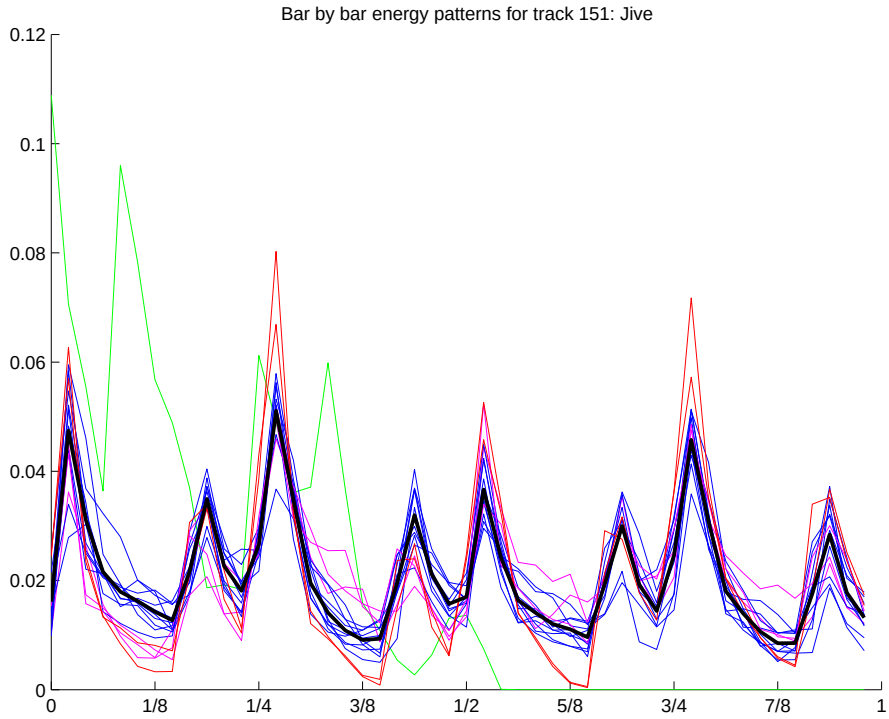


Figure 2.17: A Jive pattern showing a swing eighth note rhythm.

the k nearest instances in the training set are found, and the greatest number of these neighbours which belong to the same class determines the class of the test instance. Various values of k were used in the experiments. The standard decision tree learning algorithm, J48, was also used, as well as two meta-learning algorithms, AdaBoost and Classification via Regression. AdaBoost [FS96] runs a given weak learner (in this case J48) several times on slightly altered (reweighted) training data and combines their predictions when classifying new cases. Classification via regression (using M5P and linear regression as base classifiers) builds a regression model for each class and combines the models via voting.

Classification by Rhythmic Pattern Alone

The first set of classification experiments was performed with $p(n)$ as the feature vector for each excerpt, that is, using the rhythmic pattern alone for classification. Note that this representation is totally independent of tempo. The classification rates for various pattern representations described in subsection 2.6.3 are shown in Table 2.6. The best classification rate, 50%, was achieved using the AdaBoost classifier, with the decimated squared signal representation with $b = 120$. This is well above the baseline for classification of this data, which is 16%.

The confusion matrix for this classifier is shown in Table 2.7. Viennese Waltzes were the most poorly classified, while classification of Cha Cha pieces was the most accurate. The greatest mutual confusion was between the Waltz

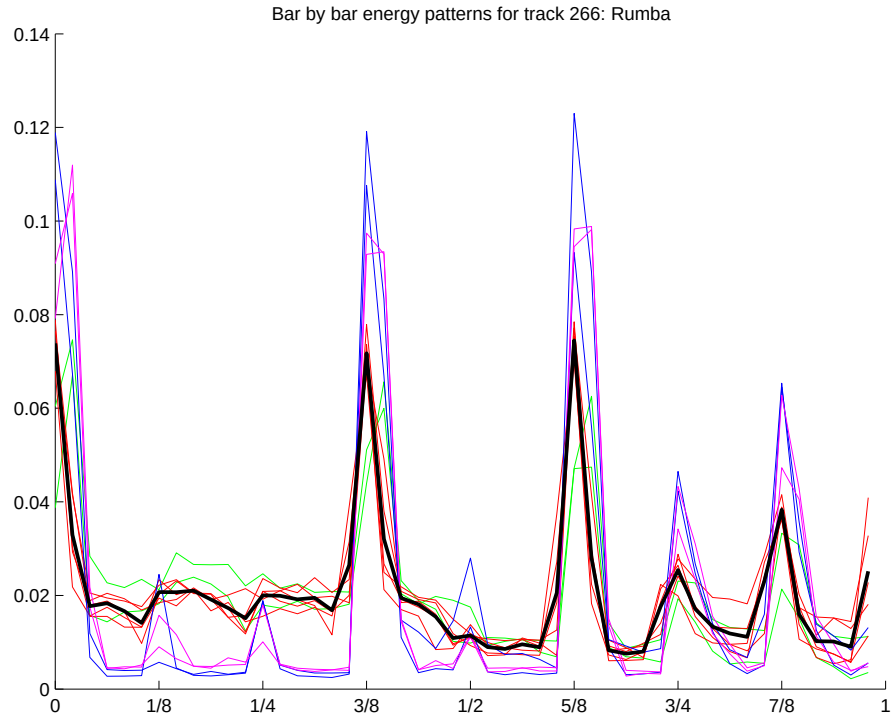


Figure 2.18: A Rumba pattern showing a strong emphasis on the 4th and 6th eighth notes. (Note that the first eighth note is at 0.)

	Resolution			
Representation	72	96	120	144
RMS ($k = 1$)	46.4%	45.7%	48.1%	45.1%
RMS ($k = 2$)	47.4%	46.0%	47.1%	46.1%
ABS	43.8%	46.1%	45.8%	46.8%
SQR	44.7%	44.7%	50.1%	45.1%

Table 2.6: Genre classification results using the rhythmic patterns alone as feature vectors. The rows are different pattern representations, and the columns are the number of points used to represent the patterns.

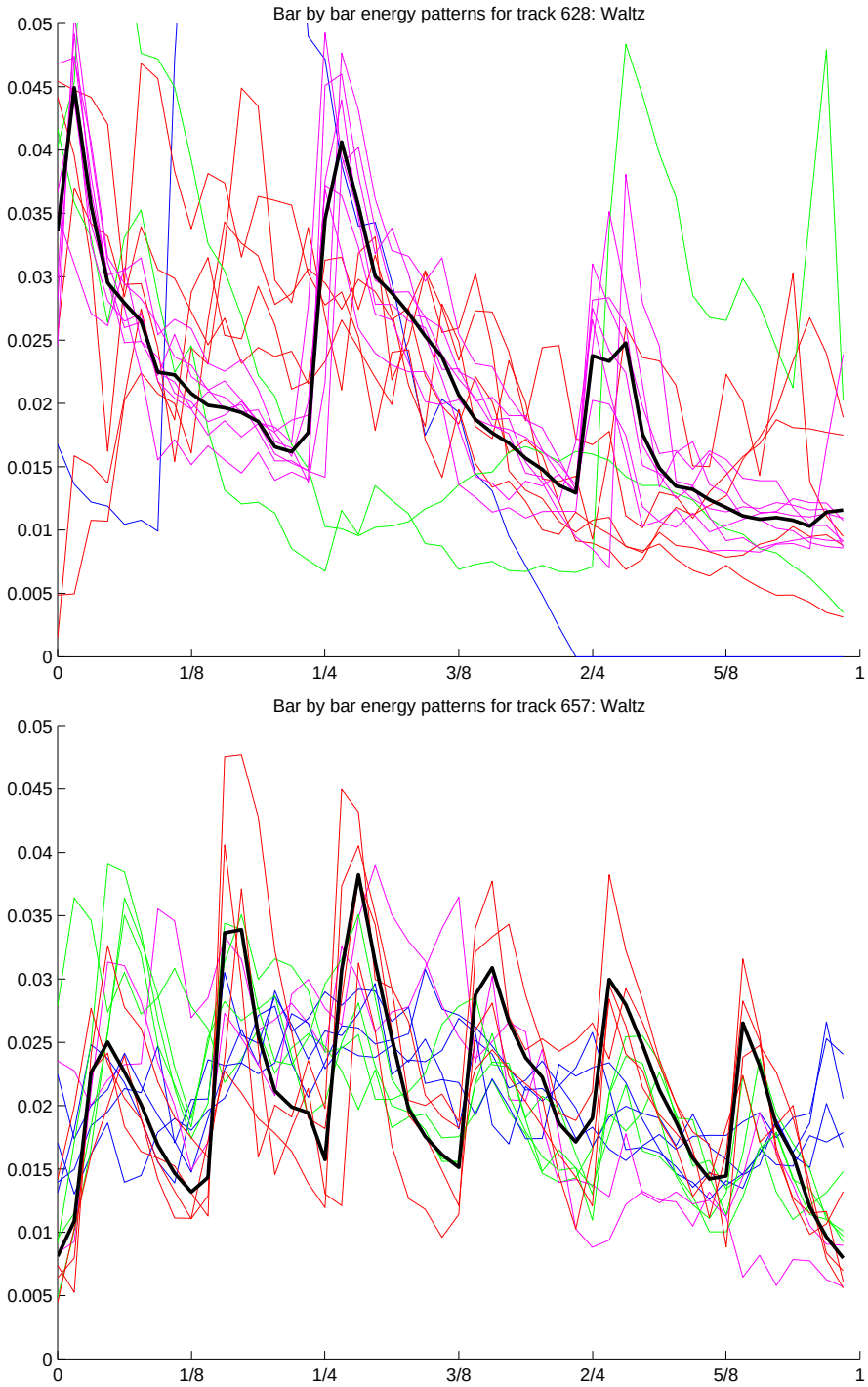


Figure 2.19: Two Waltz patterns: one in quarter notes (above), and one in eighth notes (below).

	C	J	Q	R	S	T	V	W	Rec%
C	74	6	0	14	7	7	0	3	67
J	10	23	11	1	1	10	1	3	38
Q	0	11	35	2	6	9	5	14	43
R	20	0	3	53	1	3	2	16	54
S	8	0	9	11	43	5	3	7	50
T	15	8	7	6	5	35	0	10	41
V	0	2	6	3	0	1	23	30	35
W	1	0	9	6	4	7	19	64	58
Prec%	58	46	44	55	64	46	43	44	

Table 2.7: Confusion matrix for classification based on rhythmic patterns alone. The rows refer to the actual style, and the columns the predicted style. The rightmost column shows the percentage recall for each class and the bottom row shows the percentage precision. The abbreviations for the dance styles are: Cha Cha (C), Jive (J), Quickstep (Q), Rumba (R), Samba (S), Tango (T), Viennese Waltz (V), Waltz (W).

and Viennese Waltz, which is to be expected, since they have the same metre and often use the same rhythmic patterns and instruments, and the clearly distinguishing feature, the tempo, is not encoded in the rhythmic pattern.

Calculation of Derived Features

The rhythmic patterns themselves do not contain information about their time span, that is, they are independent of the tempo. Since the tempo is one the most important features in determining dance genre [DPW03, GDPW04b], we tested classification with a combination of the rhythmic patterns, features derived from the rhythmic patterns, features derived from the audio data directly, and the tempo.

The features derived from the rhythmic patterns were: the mean amplitude of the pattern, the maximum amplitude of the pattern, the relative maximum amplitude of the pattern (maximum divided by mean), the standard deviation of the pattern amplitudes, an estimate of the metre, a syncopation factor, and the swing factor. The metre was estimated by calculating two weighted sums of the pattern, the first with higher weights around the positions of a division of the bar into 4 quarter notes (8 eighth notes), the second with the weights set for a division of the bar into 3 quarter notes (6 eighth notes). The greater of the two sums determined the metre as a binary attribute, indicating either a quadruple or ternary metre. The syncopation factor was calculated as the relative weights of the offbeat eighth notes (i.e. the 2nd, 4th, etc.) to the on-beat eighth notes. The swing factor was calculated using a pulse train of Gaussian curves spaced at quarter note intervals, correlating with the signal and finding the highest 2 peaks, which usually correspond to the positions of the quarter note and eighth note respectively. If the duration of the quarter note is q and the interval between the two peaks is r, then the swing factor s is given by:

$$s = \max\left(\frac{r}{q-r}, \frac{q-r}{r}\right) \tag{2.37}$$

	C	J	Q	R	S	T	V	W	Rec%
C	102	1	0	5	0	3	0	0	92
J	0	58	0	0	0	2	0	0	97
Q	0	0	80	0	0	0	2	0	98
R	1	1	0	92	0	4	0	0	94
S	0	0	2	0	82	0	1	1	95
T	1	0	0	2	0	83	0	0	97
V	0	0	0	0	0	0	65	0	100
W	0	0	0	1	1	0	0	108	98
Prec%	98	97	98	92	99	90	96	99	

Table 2.8: Confusion matrix for classification using rhythmic patterns and other features. Compare with Table 2.7.

If only one peak in the correlation was found, the swing factor was set to 0.

An additional feature set, containing three groups of descriptors (as described by Gouyon et al [GDPW04b]) was also used. The first group of descriptors was tempo-related features, including the measured tempo calculated from the bar length. The second group consisted of features derived from the periodicity histogram representation, and the third group of features were derived from inter-onset interval histograms. Apart from the measured tempo, all of these values were calculated automatically (see Table 2.9 and [GDPW04b] for more details).

Classification Using All Features

In the following classification experiments using all features, a classification rate of 96% was achieved with the AdaBoost classifier, using the RMS signal with $k = 2$ and $b = 72$. This is remarkable, considering that spectral features are not represented at all in the data, and there is certainly some ambiguity in the relationship between music pieces and dance styles. The confusion matrix is shown in Table 2.8. More than half (16 out of 28) of the errors are caused by confusion of Cha Cha, Tango and Rumba. From Table 2.5, we see that these styles have strongly overlapping tempo ranges and the same metre, so other features must be used to distinguish these classes.

Comparisons of classification rates with various subsets of features were performed to determine the relative contribution of each subset (see Table 2.9). The left hand column shows the results from [GDPW04b]; classification using tempo alone achieved up to 82%, classification using other features not including tempo also reached 82%, and by combining these features, a classification rate of 93% was obtained. The right hand column shows the results of adding rhythmic patterns and their derived features to the feature vectors: in each case an improvement was made, with overall classification rates improving to 84% (compared with 82%) without the tempo and 96% (compared with 93%) including the tempo. For all of these results, the rhythmic patterns were generated with the RMS filter with $k = 2$ and $b = 72$, and the AdaBoost learning algorithm was used (hence the difference from published results in [GDPW04b]).

Feature sets from [GDPW04b]	Without RP	With RP
None (0)	15.9%	50.1%
Periodicity histograms (11)	59.9%	68.1%
IOI histograms (64)	80.8%	83.4%
Periodicity & IOI hist. (75)	82.2%	85.7%
Tempo attributes (3)	84.4%	87.1%
All (plus bar length) (79)	95.1%	96.0%

Table 2.9: Comparison of classification rates using various sets of features. The columns show rates without and with the rhythmic patterns (RP) and their derived features; the rows show the different feature subsets from Gouyon et al [GDPW04b], with the number of features shown in parentheses.

2.6.5 Discussion and Further Work

It is not to be expected that a single rhythmic pattern could uniquely determine the genre of a piece of dance music. Many other features which are not represented in this work are also relevant to genre, such as the choice of musical instruments, which could perhaps be represented with standard timbral features such as MFCCs. Examination of the extracted patterns shows that some of the patterns are quite trivial, such as those which show sharp peaks on each of the quarter note beats, thus only serving to distinguish triple from quadruple metre. Nevertheless, even with these limitations, the results demonstrate that rhythmic patterns are a useful feature for classification.

The fact that only 30 seconds of each song was used may have adversely influenced the results, as many songs have an introduction which does not match the style of the rest of the piece. Because of the shortness of the tracks, it was considered better to extract only one rhythmic pattern. With longer tracks it would be worthwhile to investigate classification using multiple patterns per song. It is also expected that the statistical reliability of pattern extraction would increase with the length of the excerpt.

One restriction of the current work is that it relies on an accurate estimate of the first bar. Automatic methods of finding metrical boundaries have made great progress in recent years, but they are still far from perfect, and manual correction for very large music databases is not feasible. However the errors of such systems are not random; they belong to a very small class of possibilities: tempo errors of a factor of 2 or 3, and phase errors of half (or occasionally a third or quarter) of the metrical unit. If we allow these cases, no longer considering them as errors, the classification algorithm could possibly succeed in implicitly recognising these different cases.

Another limitation is that although we do not explicitly detect percussive onsets, the methodology assumes peaks in energy (e.g. for correlation) for extracting the patterns. This limitation is seen in the patterns extracted from Waltz and Viennese Waltz excerpts. An explicit onset detection step which includes the detection of soft (i.e. non-percussive) onsets [BS03a] could be used to alleviate this problem. Another approach would be to use features other than amplitude or energy. Paulus and Klapuri [PK02] found that the spectral centroid, normalised by the energy, provided the best feature vector for describing patterns.

The high dimensionality of the pattern vectors reduces the ability of learning algorithms to build suitable classifiers. Dimensionality reduction either by PCA or by a more explicit symbolic encoding (i.e. in musical symbols) would be a step in the direction of solving this problem. If the patterns were quantised and encoded into musical units, they could be matched to explicit patterns such as those found in instructional books. Even without such an encoding, matching in the other direction, i.e. from explicit patterns to the audio data could be performed as a method of generating further features.

A related issue that is yet to be explored is the choice of distance metrics between patterns. The Euclidean distance is not necessarily ideal, as it treats all time points independently, so that, for example, peaks which almost match are penalised as heavily as peaks which are far from being aligned.

Another avenue of further research would be to extract patterns in various frequency bands in order to detect between-instrument patterns (e.g. bass drum, snare drum, hi-hat). Alternatively, recent work on drum sound recognition [HDG03b] could be used to determine multi-dimensional rhythmic patterns. These ideas would necessitate the development of more complex methods of encoding and comparing patterns.

There are numerous other directions of possible further development. The current experiments are limited in the genres of music on which they have been performed. As other labelled data sets become available, it will be possible to test the generality of this method of pattern extraction and comparison for classification of other genres. The algorithms are general purpose; no domain specific knowledge is encoded in them. The unknown issue is the extent to which other genres are characterised by rhythmic patterns.

2.6.6 Summary

We described a novel method of characterising musical pieces by extracting prominent bar-length rhythmic patterns. This representation is a step towards building higher level, more musically relevant, parameters which can be used for genre classification and music retrieval tasks. We demonstrated the strength of the representation on a genre recognition task, obtaining a classification rate of 50% using the patterns alone, 84% when used in conjunction with various automatically calculated features, and 96% when the correct tempo was included in the feature set. These classification rates represent an improvement over previous work using the same data set [GDPW04b], and higher rates than have been published on other data sets [TC02, MB03, DPW03]. However, we acknowledge the preliminary nature of these investigations in the quest to extract semantic information from audio recordings of music.

2.7 Conclusion

In this chapter we have presented the SIMAC description scheme for rhythmic information in musical data. We started by discussing some theoretical considerations about rhythm representation from raw audio signals. Then we proceeded to exemplify our work in the area starting from the lowest possible level of rhythmically- meaningful analysis, that of onset detection.

Temporal segmentation of note events, or onset detection is at the core of most automated rhythmic analysis from raw audio. In this chapter we have discussed the main approach to onset detection using energy information, and alternative approaches relying both on the sole use of phase information and on the combined use of energy and phase in the complex domain. We demonstrate quantitatively the advantages of the latter approach.

Further to its use for onset detection, we discuss the usability of the onset detection function as a signal representation that encodes valuable information about the rhythmic contents of the signal. This is made clear in the following section, where the onset detection function is used for the task of beat tracking in music signals, in a context-dependent model that combines the bottom-up detection of beats with a model that considers the consistency of tempo in the long term. Experiments demonstrate the advantages of using this context-dependent model for rhythmic analysis.

Beat tracking is in turn used to generate estimates of the tempo of songs, the main attribute used in Section 2.5 to characterise music according to rhythm. The approach based on a combination of low-level features plus tempo, and the use of Support Vector Machines for classification, shows that tempo is important, but not sufficient and that more flexible, temporally dynamic structures are needed in order to improve classification.

A solution to this problem is proposed in section 2.6, where automatically generated rhythmic patterns, derived from beat information, are used to characterise music. This approach meets with greater success than its predecessor, and is a clear demonstration of the successful use of high-level musical descriptors for retrieval applications. It shows how the right combination of signal processing methods, machine learning and musical knowledge, can adequately characterise the underlying elements that make a good description of music signals, well beyond the purely-acoustic characterisations that have been extensively used for audio fingerprinting. It also justifies the assertion that approaches that strike the right balance between data-driven and knowledge-based techniques for description, are a way to find cracks on the "glass ceiling" usually imposed on the common approach to music classification based on purely data-driven analysis of low-level features (as seen in deliverable D341).

Most of our experiments in rhythm analysis have been limited to the use of descriptors in a way that is purely concerned with the rhythmic aspect of music. However, musical dimensions are intertwined and insight into one of the dimensions can be gained by adding information of another. In the next chapter we will see an example of an approach to harmonic representation that uses beat tracking to integrate higher-level rhythmic knowledge into the harmonic analysis. This process could be fed back to the rhythm realm, as the timing (and sequence) of chord transitions can encode important information about the rhythm of a piece. The connections to the understanding of long-term structure are also evident. Moreover, we will see in future chapters how rhythmic information can be used for more abstracts characterisations of music signals, in terms of their complexity and their similarity to other signals.

Finally, and as referred in previous sections, the issue of annotation is at the core of the design and evaluation of the approaches here presented, and that is why in Appendix A we also present a semi-automatic system for rhythmic annotation.

Chapter 3

Describing Harmony

In this report, we consider the harmony of a piece to be defined by the combination of notes, both simultaneously to produce chords, and along time to produce chord progressions. Furthermore, these chord progressions are in turn closely related to the concept of key or tonality of the music piece. Chords, their progressions, and the key are relevant aspects of music perception that can be used to accurately describe and classify music content [GH04a].

In the previous chapter, we have seen how, for the case of rhythm, it is possible to characterise music using higher-level semantic descriptors [DGW04b]. These descriptors are more readily available for rhythmic than for harmonic description, as the state-of-the-art in beat, meter tracking and tempo estimation has had more success than similar efforts on chord, melody and key estimation.

As part of the Online Music Recognition and Searching (OMRAS) project, Pickens et al [PBM+02] showed success at identifying harmonic similarities between a polyphonic audio query and symbolic polyphonic scores. The approach relied on automatic transcription, a process which is partially effective within a highly constrained subset of musical recordings (e.g. mono-timbral, no drums or vocals, small polyphonies). To effectively retrieve despite transcription errors, all symbolic data was converted to harmonic distributions and similarity was measured by computing the distance between two distributions over the same event space. This is an inefficient process that goes to the unnecessary step of transcription before the construction of an abstract representation of the harmony of the piece.

In SIMAC, we adopt the approach where we try to describe the harmony of the piece, without attempting to estimate the pitch of notes in the mixture. By avoiding the transcription step, we also avoid its constraints, allowing us to operate on a wide variety of music. However, this requires using a set of low-level features that is able to emphasise the harmonic content of the piece, such that this representation can be exploited for further, higher-level, analysis.

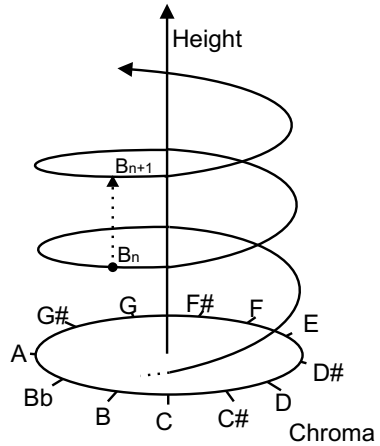


Figure 3.1: The Pitch Helix. Note B_{n+1} is an octave above note B_n.

It is possible to model our perception of pitch using a helix, as shown in Figure 3.1, where the importance of the octave interval is preserved with notes an octave apart directly above one another [She64]. From this helical repre-

sentation it can be seen that pitch can be given two attributes: *Pitch Height*, moving vertically in octaves and *Chroma* (or *Pitch Class*), its rotation on the helix. Two pitches an integer number of octaves apart will share the same rotation on the Chroma Circle shown at the base of the helix in Figure 3.1. The musical intuitiveness of the chroma makes it an ideal feature representation for note/chord events in music signals. A temporal sequence of chromas results in a time-frequency representation of the signal known as chromagram. The Chromagram, also known as Harmonic Pitch Class Profile (HPCP [GH04c]), is a vector representation of the Chroma Circle that can be automatically extracted from the raw audio signal.

In the following, we will explain how the Chromagram can be generated and used for the task of tonality estimation from music signals. Then we will concentrate on the issue of chord estimation, first focusing on the accurate generation and processing of the feature set and a simple template-based model of chords, and then proposing a more complex model that explicitly includes musical knowledge about the nature of chords and chord progressions. Details about the generation of an annotated chord database can be found in Appendix refapp:chordann. Finally we present a new trend of research, started under the SIMAC project, for the object-oriented analysis of pitched components in raw audio signals. This form the basis of current work on the segregation of melodic lines from polyphonic recordings.

3.1 Tonality estimation

The tonality and other tonal attributes have great relevance on the appreciation of music. Previous studies have tried to relate those attributes with the induction of mood in listeners. Furthermore, and as will be reported in chapter 8, we can find certain similarity between different excerpts sharing the same tonality. Listeners are sensitive to key changes, which are also related to rhythm, structure, style and mood. Key changes can be used, for instance, as cues about the structure of a song, or as features to query for matching pieces in a database.

In western music, the term key (or tonality) is usually defined as the relationship between a set of pitches having a tonic as its main tone, after which the key is named. A key is then defined by both its tonic (also called key note, for example: *A*) and its mode (e.g. minor). The tonic is one in an octave range, within the 12 semitones of the chromatic scale (e.g. *A, A#/Bb, B, C, C#/Db, D, D#/Eb, E, F, F#/Gb, G*). The mode is usually minor or major, depending on the used scale. The major and minor keys then rise to a total set of 24 different tonalities.

In this section we present an approach to the automatic computation of the tonality from raw audio signals containing polyphonic music. The method is based on a tonality model that has been established after perceptual studies, and uses some musical knowledge to estimate the global key note and mode attached to a certain audio segment. In the following we describe and evaluate the method, present some results and discuss our findings.

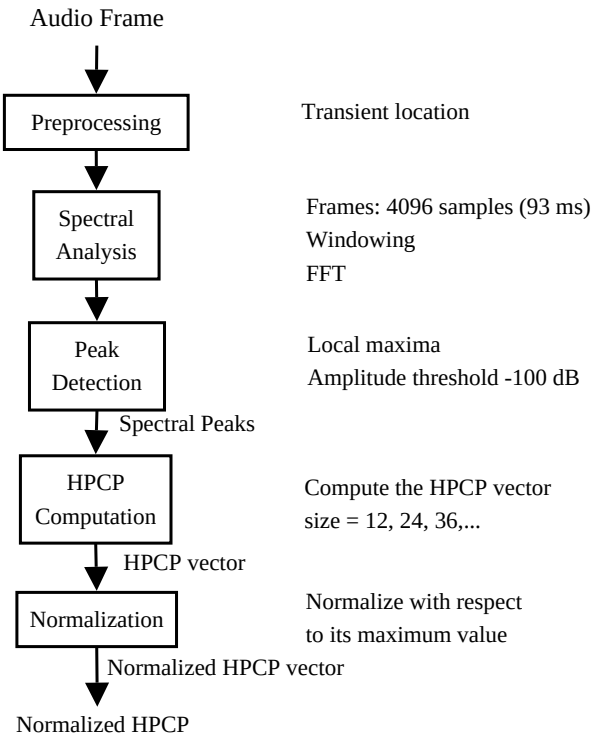


Figure 3.2: Diagram for HPCP Computation

3.1.1 The feature set: the Harmonic Pitch Class Profile

The Harmonic Pitch Class Profile (HPCP) measures the intensity of each of the twelve semitones of the diatonic scale, by mapping each frequency bin of the spectrum to a given pitch class. This vector is computed for each analysis frame roughly as follows: first we perform the spectral analysis of the input signal; then we calculate the spectral peaks, or local maxima, of the spectrum; finally, we "fold" the position of the peaks to one octave, assigning each peak's energy to its corresponding pitch class. The complete process is depicted by the diagram in Figure 3.2.

The following constraints apply: the HPCP only uses the spectral peaks in a certain frequency band to compute its intensity; second, we introduce a weight into the feature computation; third, we use high resolution in the HPCP bins (decreasing the quantisation level to less than a semitone). This can be summarised as follows:

$$HPCP(n) = \sum_{i}^{nPeaks} w(n, f_i) a_i^2 \tag{3.1}$$

where $n = 1, ..., size$, a_i is the linear magnitude of the ith peak, and f_i is the frequency value of the ith peak. $i = 1, ..., nPeaks$, where $nPeaks$ is the number of spectral peaks that we consider, n is the HPCP bin, $size$ is the size of the HPCP vector (i.e. number of bins: 12, 24, 36, or other), and $w(n, f_i)$ is the

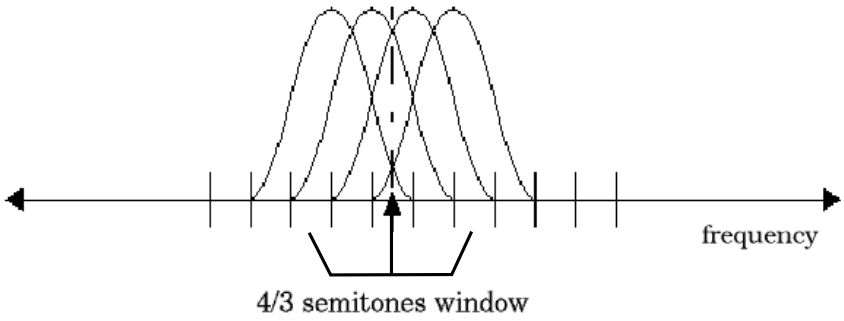


Figure 3.3: Weighting function

weight of the frequency f_i when considering the HPCP bin n.

Weighting Function

Instead of contributing to a single bin (e.g. the closest one), each frequency f_i contributes to the HPCP bin(s) that are contained in a certain window around its frequency value, as shown in Figure 3.3. For each of those bins, the contribution of the peak i with frequency f_i (which is the square of the peak linear amplitude a_i^2) is weighted using a $\cos^2$ function around the frequency of the bin. The value of the weight depends on the frequency distance between f_i and the centre frequency of the bin n, f_n, measured in semitones.

This weighting procedure (as illustrated in Figure 3.3) minimises the estimation errors that we find when tuning is not perfect and/or inharmonicity is present in the spectrum. These factors can induce errors when mapping frequency values into HPCP bins.

Normalisation

For each analysis frame, the HPCP values are normalised with respect to its maximum value, in order to store the relative relevance of each of the HPCP bins.

$$HPCP_{normalized}(n) = \frac{HPCP(n)}{Max_n(HPCP(n))} \tag{3.2}$$

where $n = 1, ..., size$.

3.1.2 Estimating tonality and defining key profiles

As shown in Figure 3.4, we compute a correlation of the HPCP vector with a matrix of HPCP profiles corresponding to the different keys K, with size $2 \times 12 \times size$. The maximum correlation value corresponds to the estimated key note and mode (represented by the indexes i_{max} and j_{max}). We use the maximum correlation value $R(i_{max}, j_{max})$ as a measure of the *tonalness*, the degree of tonality or key strength:

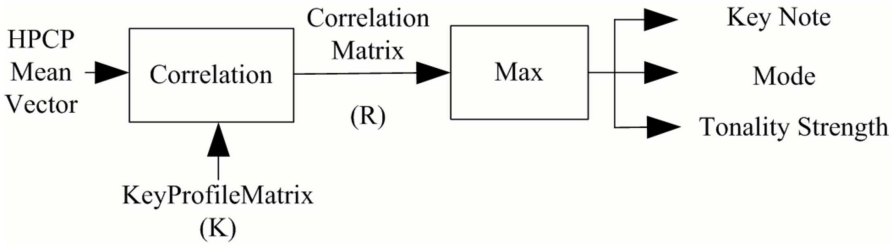


Figure 3.4: Block Diagram for Key Computation using HPCP

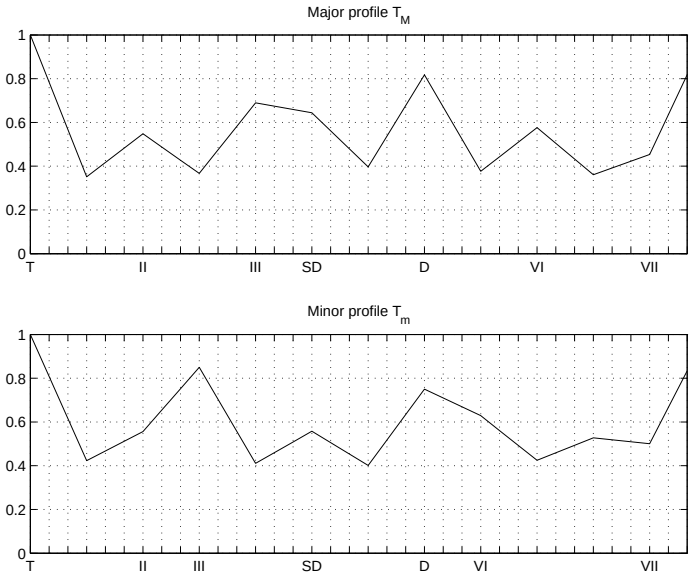


Figure 3.5: Major and minor profiles, T_M and T_m, as proposed by Krumhansl (1990) and normalized by their maximum values

$$R(i_{max}, j_{max}) = max_{i,j}(R(i,j)) \tag{3.3}$$

To construct the key-profile matrix, we use a method based on the model proposed by Krumhansl and Schmuckler used for key estimation from MIDI representations (Krumhansl 1990). We have adapted this technique in two aspects: first to work with HPCP instead of note-duration values, and second to consider polyphony instead of melodic lines. In polyphonic music, more than one note can sound at the same time. Major and minor Krumhansl profiles, T_M and T_m respectively, are represented in Figure 3.5.

The tonal hierarchy within a melodic line should be maintained in a polyphonic situation, where the melodic line is converted into a chord sequence. We should then consider all the chords containing a given pitch class when measuring the relevance of this pitch class within a certain key. We compute the probability of finding a given pitch class in a given major key $T_{Mp}(i)$, as the weighted sum of the probability of finding each of the chords (of a major key)

to which this pitch class belongs:

$$T_{Mp}(i) = \sum_{j=1}^{12} \alpha_M(i,j) T_M(j) \tag{3.4}$$

where $i = 1, ..., 12$. In the same way, we compute the probability of finding a given pitch class in a given minor key $T_{mp}(i)$, as the weighted sum of the probability of finding each of the chords (of a minor key) to which this pitch class belongs:

$$T_{mp}(i) = \sum_{j=1}^{12} \alpha_m(i,j) T_m(j) \tag{3.5}$$

where $i = 1, ..., 12$. These equations can be also expressed as follows:

$$T_{Mp} = \alpha_M T_M^t \tag{3.6}$$

$$T_{mp} = \alpha_m T_m^t \tag{3.7}$$

where T_{Mp}, T_M, T_{mp}, and T_m are 1×12 vectors, and α_M, α_m are 12×12 square matrices . Within a major key, $\alpha_M(i,j)$ represents the weight of the pitch class i when considering the chord whose root is pitch class j. $\alpha_m(i,j)$ represents the weight of the pitch class i when considering the chord whose root is pitch class j within a minor key. After performing some comparative tests (see Section 3.1.4), we consider only the three main triads of the key as the most representative ones: tonic ($i = 1$), dominant ($i = 8$), and subdominant ($i = 6$).

Finally, some modifications are made to the profiles to take into account the fact that we work with audio features (HPCP values) instead of MIDI. The spectrum of a note is composed of several harmonics, whose frequencies are multiples of the fundamental frequency (f, $2f$, $3f$, $4f$, etc.). Then, when a note is played, the intensity increases at the frequency of the different harmonics. This fact appears in the HPCP values, where the value increases for different indices.

The profile values for a given pitch class i ($T_{Mp}(i), T_{mp}(i)$) are equal to the sum of contributions of all the pitch classes containing i as a harmonic pitch class. That means that each note of the chords under consideration contributes to the profile values of its i_n harmonics ($n = 1, 2, ..., nHarmonics$). We make the contribution decrease with the frequency by multiplying this contribution with a function with a certain decay s, in order to simulate that the spectrum amplitude decreases with frequency.

The final profiles T_{Mp} and T_{mp} are represented in Figure 3.6.

3.1.3 Examples and discussion

In this section, we present some examples of how the algorithm works. Figure 3.7 shows the results of the analysis of a polyphonic excerpt in C major. Average HPCP values, as well as correlation with major and minor models, are shown. In the HPCP profile, we can identify peaks in the tonic C and dominant G notes, as well as in the major third E. The correlation shows a maximum value of 0.84 in C major. Other peaks appear a 5th above within the circle of fifths (which is G major and its relative minor E minor), as well as in C minor, which shares the dominant and subdominant chords (considering harmonic minor scale). In both

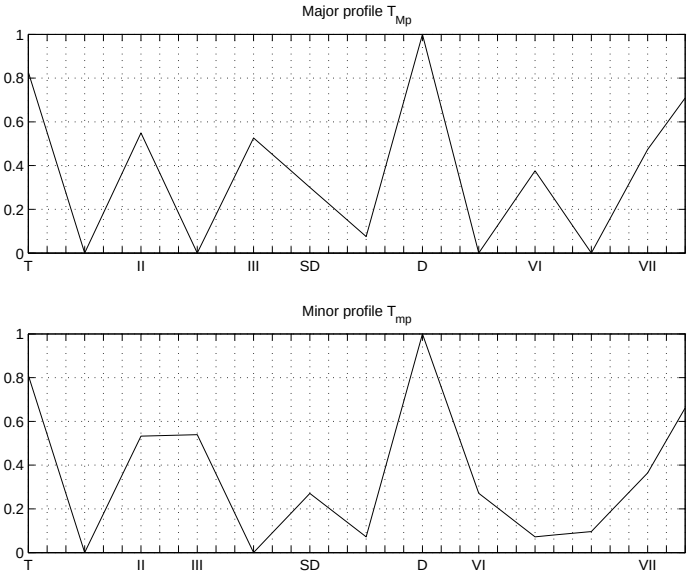


Figure 3.6: T_{Mp} and T_{mp} Profiles

examples we can check that the features represent some aspects of tonal content. These aspects are considered in other models that represent the relationships within keys (Temperley 1999, Chew 2000).

Figure 3.8 shows the results of the analysis of a polyphonic excerpt in F♯ minor. As in the last example, mean HPCP values, as well as correlation with major and minor models are shown. In the HPCP profile, we can identify peaks in the tonic F♯ and dominant C♯ notes, as well as on the minor third A. The correlation shows a maximum value 0.79 in F♯ minor. Other peaks appear a 5th below within the circle of fifths (which is D major and its relative minor B minor), as well as in F♯ major, which shares the dominant chord (considering harmonic and melodic minor scales) and the subdominant chord (considering melodic minor scale).

Figure 3.9 shows the results of the analysis for a polyphonic percussion excerpt, where we cannot identify any key. In the HPCP profile there is no clear peak, while the correlation peaks barely exceed 0.5. In this case, the low correlation values indicate that the key is not well defined.

3.1.4 Evaluation

For evaluation purposes, we have set up a small test database with 35 samples (passages of polyphonic musical pieces) of different styles, and in various key and modes, labelled by hand in terms of key. Each excerpt has a constant tonality. The results are presented in Table 3.1 and in Figure 3.10. For this test database, we have tested different configurations in order to measure the improvements of the following aspects:

1. Comparison of PCP (Fujishima 1999) or HPCP for key estimation. The original PCP algorithm was modified to compute 36 values instead of 12, in order to get a better frequency resolution.

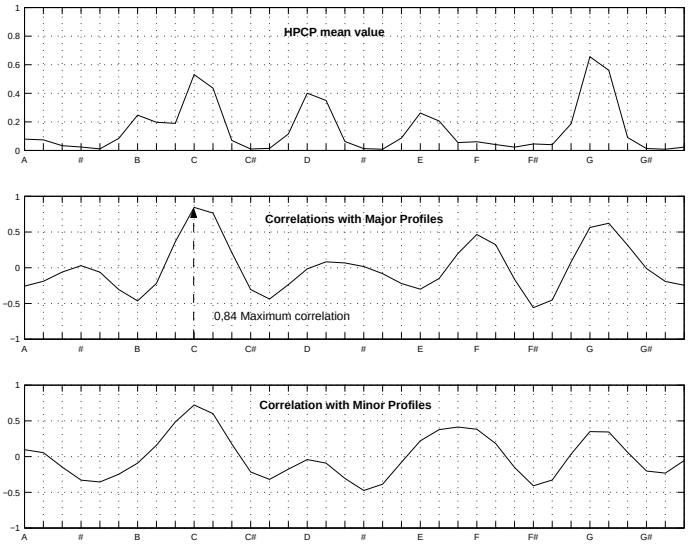


Figure 3.7: Example of HPCP Profile for a C Major key Audio excerpt

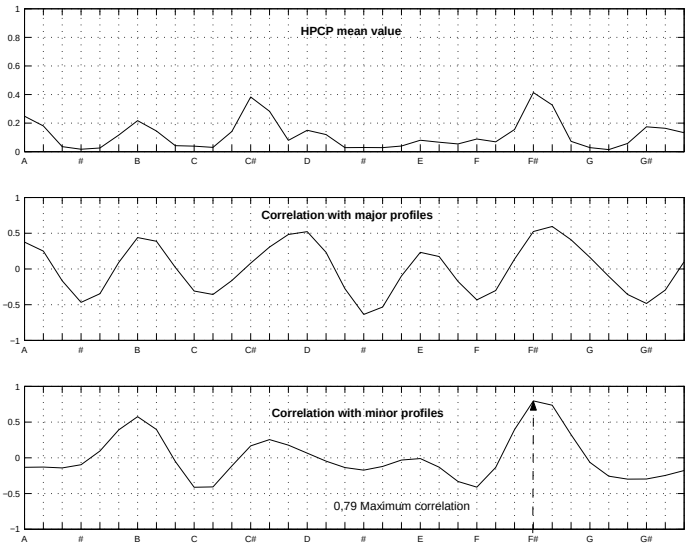


Figure 3.8: Example of HPCP Profile for an F♯ Minor Key Audio Excerpt

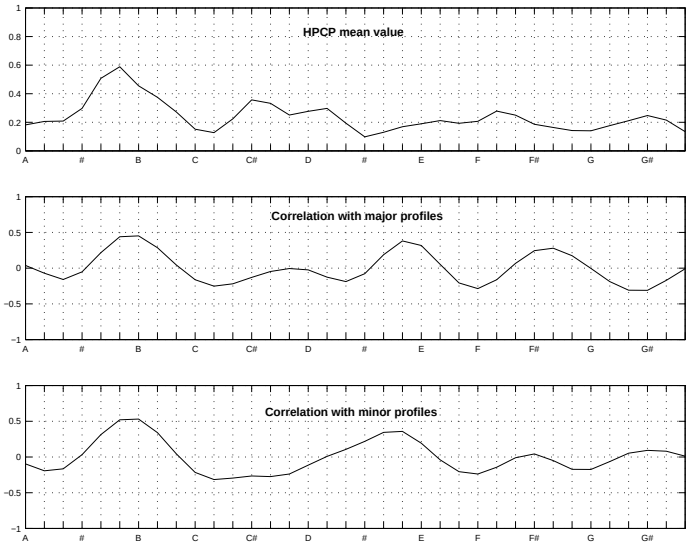


Figure 3.9: HPCP Profile for a Percussive Sound (No Clear Tonality)

2. Comparison of the original profiles T_M and T_m (Krumhansl 1990) against the proposed $T_{Mchords}$ and $T_{mchords}$, built either considering all the chords of the key or only the three most relevant chords (i.e. tonic, dominant, and subdominant chords).

Results are presented in Table 3.1, where several evaluation metrics are shown. The first ones are the percentage of correct key estimation (both tonic and mode), the percentage of correct mode estimation and the percentage of correct key note estimation. These three metrics evaluate how the algorithm performs for key, mode, and tonic (key note) estimation. Percentages are also shown for the cases where the error is only in mode estimation (i.e. confusions with major/minor relatives) and when the error is only in key-note estimation. Finally, we show the percentage of the items where both key note and mode were not correctly estimated.

We observe that there are significant improvements when using HPCP and modified Krumhansl profiles, obtaining a 75.8% of correct estimation, 90.9% of correct mode estimation, and 78.8% of correct key note estimation when using the three main chords of the major and minor keys. When the window size is increased to 371 ms (16384 samples), the performance also increased to 81.8%. This performance was achieved when considering the main chords to build the profiles, and the same results were obtained using HPCP and PCP ($size = 36$ bins). That suggests that HPCP is more robust than PCP with smaller window sizes, i.e. lower frequency resolution for FFT.

We also evaluated the performance of the algorithm on a database of 883 classical music pieces segmented by track and labelled by title (e.g., *Mozart, Flute Concerto No 1 K313 G Major Andante non troppo*). They include composers such a Mozart, Chopin, Scarlatti, Bach, Brahms, Beethoven, Handel, Pachelbel, Tchaikovsky, Sibelius, Dvorak, Debussy, Telemann, Albinoni, Vivaldi, Pasquini, Glenn Gould, Rachmaninoff, Schubert, Shostakovich, Haydn,

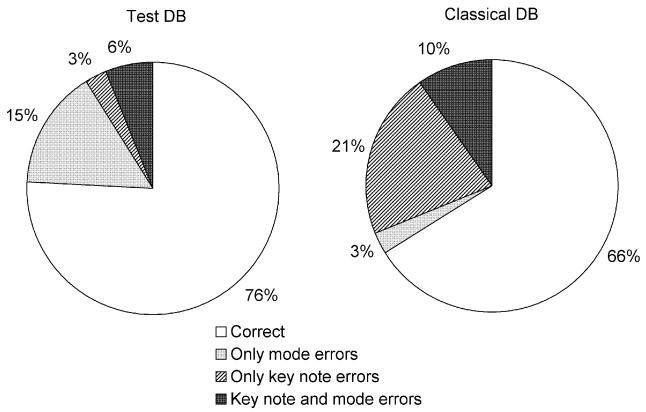


Figure 3.10: Evaluation results

Benedetto, Elgar, Bizet, Listz, Boccherini, Ravel, Debussy, etc. We also included some jazz versions of classical pieces (e.g. Jacques Lousier, The Swingle Singers). Most of the included pieces were first movements (in the case that the piece is a multi-movement form as a sonata or symphony). All the key note and mode annotations were taken from the FreeDB database (FreeDB 2004). Some additional manual corrections were done because of incorrect FreeDB metadata, although systematic checking was not performed. We assumed that the key is constant within the whole piece. That means that the modulations we find do not modify the overall key of the piece. We considered the parameters that yield better results with the small test database: HPCP and three main chords, using a window size of 4096 samples (93 ms). The results are presented in the Table 3.2 and in Figure 3.10.

Correct estimation is obtained for 66.1% of the items, and mode is estimated correctly for 87.5% of the pieces. The majority of errors occur when the mode is correctly estimated but not the key note.

In Figure 3.11, we present the confusion matrix for the estimation. Rows represent the correct tonality, and columns the estimated tonality. We find that 5% of the errors were due to minor/major relatives, 11% to tuning errors (one semitone higher or lower), 14% of the errors estimated the upper 5th within the circle of fifths, and 40% a 5th lower within the circle of fifths. The confusion matrix reveals the need to consider, both in the key model and in the analysis of the correlation functions, some aspects of perceptual similarities, already pointed out in the literature (Temperley 1999, Chew 2000, Janata et al. 2002).

3.1.5 Conclusion

3.2 Chord Identification

On the previous section, we have seen how chroma-based features can be used to describe the tonality of the song. However, we need to assume that the tonality is constant along the analysed piece, thus we have no way to characterise the local changes that may occur within. In this section we focus on the local analysis of chords, in order to provide a bottom-up method of describing the

Percentage	PCP Krumhansl	HPCP Krumhansl	HPCP All Chords	HPCP Main chords	PCP Main chords
Correct	36.4	51.5	**75.8**	**75.8**	42.4
Correct mode	69.7	75.8	87.9	**90.9**	66.7
Correct key note	48.5	54.5	75.8	**78.8**	51.5
Only mode errors	33.3	24.2	12.1	15.2	24.2
Only key note errors	12.1	3	0	3	9.1
Key note and mode errors	18.2	21.2	12.1	6.1	24.2

Table 3.1: Results of Evaluation of a Small Test Database (%)

Table 3.2: Results of Evaluation of a Classical Test Database

	percentage
Correct	66.1
Correct mode	87.5
Correct key note	68.9
Only mode errors	2.72
Only key note errors	21.4
Key note and mode errors	9.74

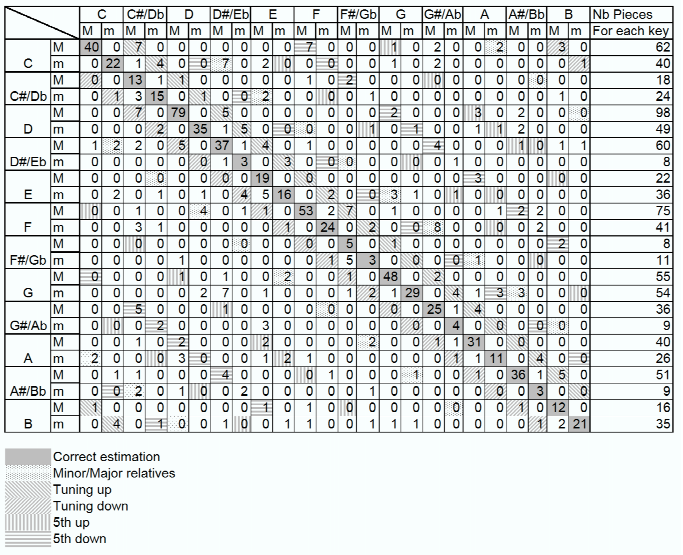


Figure 3.11: Confusion Matrix for the Classical Database. Rows Represent the Correct Key Whereas Columns Represent the Estimated One. The Number of Confusions is Written in the Corresponding Cell.

harmony of the piece.

When two or more notes are played simultaneously the result is a musical chord. The type of chord produced is dictated by the musical intervals between its component notes. There are many different ways in which a particular chord may be played, or *voiced*. Although different voicings of the same chord may not sound identical to each other, we still perceive them as being musically the same because they contain the same constituent pitch classes [Tay89]. Therefore, a chromagram is a suitable low-level feature for chord identification.

Although a 12-element chromagram (one element per semitone) is musically intuitive, it is not sufficient to directly calculate a chromagram with only semitone resolution. This is because the tuning pitch of musical instruments can vary from recording to recording. It cannot be assumed that orchestras and bands will always be tuned to concert pitch with A4=440Hz. It is possible, and quite common in 1960s pop music, for tuning to be as much as a full quarter-tone away from concert pitch, so a higher resolution is necessary.

In the approach presented here we derive a 36-bin HPCP from a Constant-Q spectral transform [Bro91b]. A peak-picking algorithm is applied to the HPCP

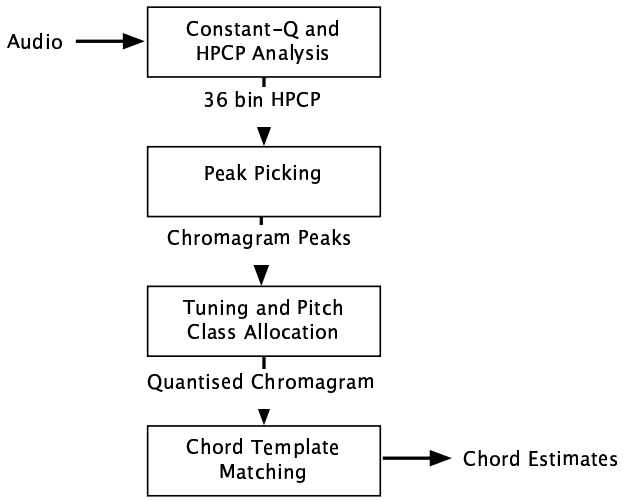


Figure 3.12: Flow Diagram of System

and the precise positions and amplitudes of peaks are then found using quadratic interpolation. A tuning algorithm is then applied to these peaks to find the tuning pitch for the audio extract. A histogram of the distribution of the peak positions is taken across one semitone width, which is equivalent to three chroma bins. The maximum point in the histogram gives the semitone centre tuning value. Once this tuning value is calculated, the positions of the boundaries between pitch classes can easily be found and each peak can then be assigned to a pitch class. The result is a 12-bin, semitone-quantised chromagram that can be used for further harmonic analyses.

For the task of chord identification, each frame of the quantised chromagram is compared with a set of chord masks and the closest match is then recorded as the chord estimate for that frame. This chord identification technique is similar to that described by Cremer and Derboven [CD04] and also Pardo and Birmingham [PB02].

A flow diagram of the system is shown in Figure 3.12 and a detailed description of each part is given in the following. Experimental results are presented in Section 4.2.6 and further work is discussed in Section 3.2.6.

3.2.1 Constant-Q Analysis and HPCP

The first stage of the system is the Constant-Q spectral analysis [Bro91b]. This is a logarithmic frequency analysis implemented using code based on the efficient algorithm described in [JCB92]. The audio files are downsampled to 11025Hz and we calculate a 36 bins-per-octave transform between f_{min}=96Hz and f_{max}=5250Hz. This gives the k^{th} bin centre frequency as:

$$f_k = (2^{1/36})^k f_{min} \tag{3.8}$$

The 36 bins-per-octave resolution ensures that it is possible to distinguish between adjacent semitone frequencies regardless of the tuning of the recording [GH04c, PBO00]. To obtain a Constant-Q analysis with these parameters, a window length of 8192 samples is required. This is approximately 0.74s which is a relatively long analysis window in terms of musical harmony. Thus, to improve time resolution, frames are overlapped by $\frac{1}{8}$th of a window length giving a time resolution of 0.093s per frame.

A 36-bin HPCP is now calculated from the Constant-Q spectrum vector, C. The HPCP vector for a frame is given by:

$$\text{HPCP}_b = \sum_{m=0}^{M} |C_{(b+36m)}| \qquad 1 \leq b \leq 36 \tag{3.9}$$

where M is the number of octaves spanned by the Constant-Q spectrum and b is the HPCP bin number.

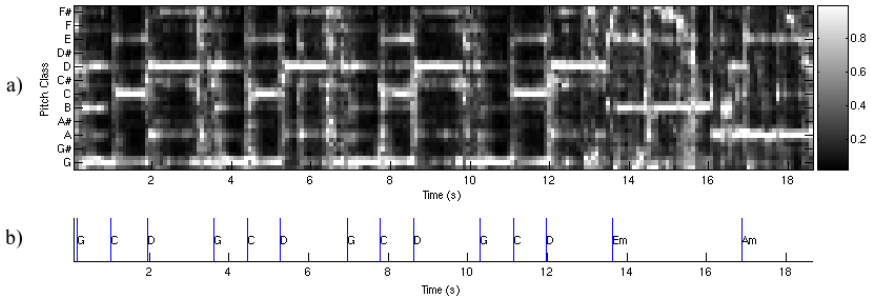


Figure 3.13: a) 36-bin HPCP and b) hand-labelled chords for the opening bars of *Another Crossroads* by Michael Chapman

Figure 3.13 shows the 36-bin HPCP for the opening bars of *Another Crossroads* by Michael Chapman. The shapes of different chords can be seen quite clearly at the start of this example. From 0 to 1 seconds the bins around G, B and D have high intensity showing a G major chord. This is followed by a C chord (C, E and G) then a D chord (D, F♯ and A). The pattern then repeats 3 more times before changing to the Em chord just before 14 seconds.

3.2.2 Peak-Picking and Interpolation

A peak-picking algorithm is now applied to the HPCP vector. When the peak bins have been identified, a quadratic interpolation step is performed to obtain values for the position and magnitude of the peaks.

3.2.3 Tuning Algorithm

Using the accurate positions and magnitudes of peaks in the chromagram it is possible to determine a tuning centre for the pitch classes. The approach taken to achieve this is to examine the distribution of peak positions across the width of one semitone (3 HPCP bins).

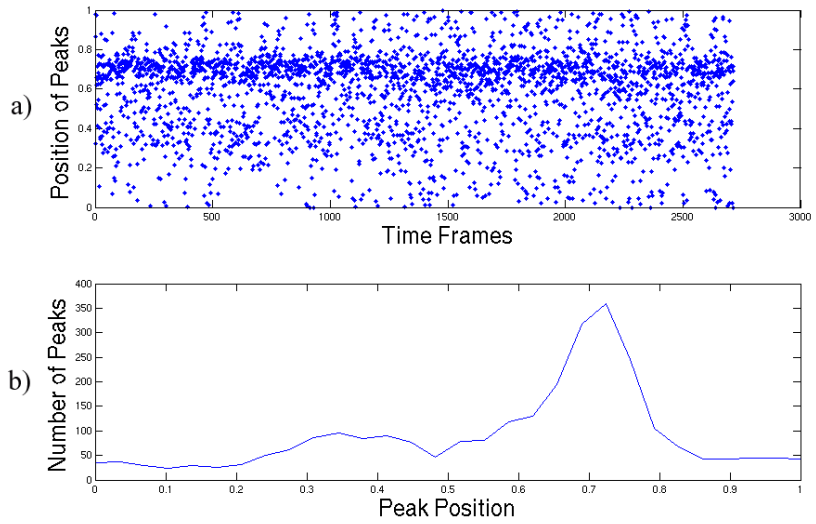


Figure 3.14: a) Tuning algorithm distribution of peaks and b) Histogram of peaks from the HPCP in Figure 3.13

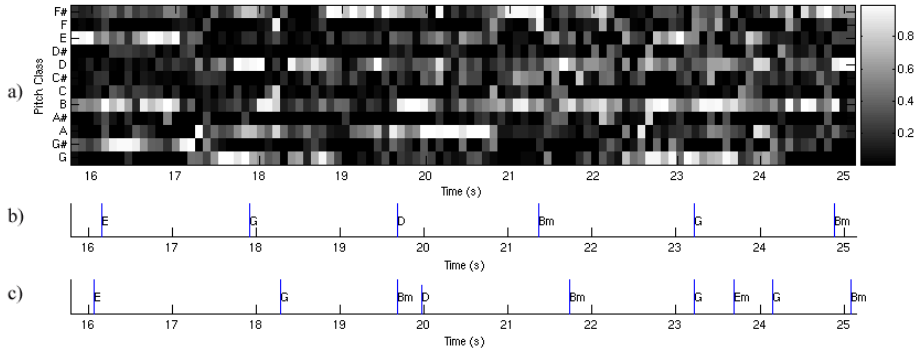


Figure 3.15: Plots for an excerpt from The Beatles *Eight Days a Week*. a) the Semitone-quantised Chromagram, b) hand annotated chord symbols and c) chord symbols produced by the system

A histogram is calculated over the length of the audio file and the maximum value in that histogram is taken as the centre tuning value (see Figure 3.14). With this value calculated, the positions of boundaries between the semitones are known so it becomes possible to allocate each peak to one of the twelve pitch classes. The result is a 12-bin semitone-quantised chromagram which is then useful for musical analysis. The plot in Figure 3.15(a) shows the quantised chromagram for an excerpt from *Eight Days a Week* by the Beatles.

3.2.4 Chord Identification

A chord identification algorithm is applied to each frame of the quantised chromagram to give a sequence of chord estimates.

Our identification algorithm uses chord templates in a similar way to that described in [CD04, PB02]. The chord templates are simple 12-element bit masks with 1 where a chord note is present and 0 elsewhere. In this way, the mask for a major[1] chord becomes 100010010000. The system recognises the four types of triad chords: major, minor, augmented and diminished. Each mask has twelve possible rotations (one for each pitch class) so a total of 48 different chords can be distinguished.

The current frame of the chromagram is multiplied by each of the twelve rotations of the four bit masks. The highest output from the multiplication gives the closest match for that frame and is recorded as the chord estimate.

Transient and noise-like signals, such as drum sounds, can corrupt short numbers of frames in the chromagram since they obscure the harmonic signal components. Applying a finite impulse response lowpass filter to the chromagram across a number of frames prior to the template matching stage reduces the effects of such signals on the results. A median filter applied to the output sequence has also been found to improve results because chord changes that are very short in duration are musically unlikely to occur.

Figure 3.15(b) and 3.15(c) show the hand-labelled chords compared to the output estimates from the system for the excerpt from *Eight Days a Week*. Figure 3.16 shows a plot of the system's output estimates for the same exerpt of *Eight Days a Week* compared with the hand labelled chord values.

3.2.5 Evaluation

To test the accuracy of the sequence of estimates produced by the algorithm, we compare against a sequence derived from a hand-labelled file. The test set of 28 pieces used in experiments has been taken from the Beatles albums *Please Please Me* and *Beatles for Sale*. Details of the methodology used for the annotation of the database can be found in Appendix B. For the future, we are considering extensions to the annotated database by using the approach in Appendix **??**.

Initial results from our experiments, shown in table 3.2.5 are encouraging. They compare favourably with results of work by Sheh and Ellis in [SE03b] who use trained Hidden Markov Models to recognise chords on a similar test set from the Beatles back catalogue. In their paper, they test their system on the two Beatles songs *Every Little Thing* and *Eight Days a Week* and show chord labels for the same short excerpt of *Eight Days a Week* that is shown here in Figure 3.15. The results from our system for these two songs are shown at the top of table 3.2.5 .

The results for the two Beatles albums are very different. The system correctly identifies chords in an average of 70.8% of frames for *Beatles for Sale*, their fourth album, whereas for their first album *Please Please Me* it scores 53.9%. This difference may be due to the difference in performance and production style between the two albums. The songs from the earlier album are arranged mainly for two electric guitars, drums and bass with keyboards and harmonicas

[1] A major chord comprises the tonic, or root note plus a major third and a perfect fifth [Tay89]

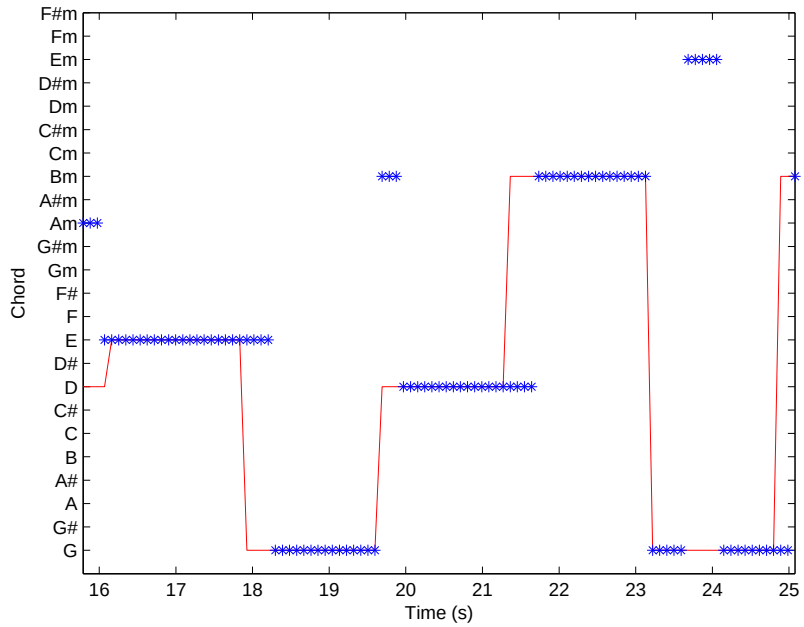


Figure 3.16: Plot of chord estimates (shown as asterisks) for excerpt from *Eight days a week* with hand annotated chords (solid line).

featuring as solo instruments. The later album is more heavily produced with keyboard instruments and acoustic guitars often adding more to the harmonic texture of the arrangements.

The most common errors that occur are confusion between the desired chord and chords which it is harmonically closely related to. Confusion between the desired chord and its Dominant (the most strongly related chord in Harmony) accounts for a high number of the errors. This may be due to identification errors where chords change because of the high proportion of Tonic-Dominant and Dominant-Tonic transitions in popular music.

In the example in Figure 3.15 from *Eight Days a Week*, two of the short chord identification errors are caused by a common confusion between closely related relative major and minor chords (Bm instead of D and Em instead of G). In the case of the confusion between G and Em, this can be further explained by the fact that the actual chord being played is a G6, which may also be written as Em/G because it contains the same notes. The only difference between these two chord labels is the musical key context in which they are placed so both may be argued to be correct instantaneous estimates.

3.2.6 Discussion and Further Work

The chord templates that the current system uses are bit masks based on known chord intervals. These do not model the other harmonic information that will be present in a chromagram for a chord played on real instruments. In future work it is possible that templates derived from chromagrams of real chord training sets may perform better than these simple masks.

Due to the circular nature of the chroma representation, the system currently has no way to determine the bass note of the chord. The bass has an important effect on the harmonic function of a chord so a separate bass tracking algorithm would be very useful in helping to make better chord identification decisions.

Some of the errors in the output sequence which are caused by transients might be reduced more effectively by the application of some preprocessing steps to the audio signal. Transient/Steady-State separation such as that described by Duxbury [Dux04] might give a cleaner 36-bin HPCP and so make the chord identification more reliable.

Four chord types are modeled by the current system. In future implementations this will be extended to include other chord types which can be modeled with the chromagram, including the 7th chords.

Finally, and perhaps most importantly, the system as it stands calculates frame-by-frame chord estimates. Apart from applying a median filter to the output sequence no post-processing is performed on the data. Results suggest that high-level contextual models of inter-key and inter-chord relationships and harmonic progression may be used to improve segmentation and decisions between chords. We explore this approach in the following section.

3.3 An Alternative Representation for Harmonic Content

For the semantic description of harmonic content from raw audio, we do not need to perform a formal harmonic analysis, but to produce a robust and consistent

harmonic description that is useful for similarity-based applications.

The method in section 3.1 correlates chromagrams with cognition-inspired models of key profiles to estimate the overall key of music signals [GH04b]. Similarly, the approach in section 3.2 correlates tuned chromagrams with simple chord templates for the frame-by-frame estimation of chords in complex signals [HS05]. While differing in their goals, these studies identified the lack of contextual information about chord/key progressions as a weakness of their approaches, as at the level of analysis frames there are a number of factors (e.g. transients, arpeggios, ornamentations) that can negatively affect the local estimation.

In this section we present an approach that combines a chroma-based representation and a hidden Markov model (HMM) initialised with musical knowledge and partially trained on the signal data. The output, which is a function of beats (tactus) instead of time, represents the sequence of major and minor triads that describe the harmonic character of the input signal [BP05].

In the following we briefly review previous work on this area; then give details about the construction of the feature vector; we explain the used model and justify our initialisation and training choices; we then proceed to evaluate the representation against a database of annotated pop music recordings; and finally present our conclusions and directions for future work.

3.3.1 Background

We are by no means the first to use either chroma-based representations or HMMs for automatically estimating chords, harmony or structure from audio recordings.

In their research on audio thumbnailing, [BW01] found that the structure of a piece, as seen by calculating a similarity matrix, is more salient when using beat-synchronous analysis of chromas. Longer analysis frames help to overcome the noise introduced by transients and short ornamentations. However, this solution still does not make use of the fact that in a harmonic progression certain transitions are more likely to occur than others.

An alternative way of embedding the idea of harmonic progression into the estimation is by using HMMs. The work by [RS03] is a good example of successfully using HMMs for harmonic analysis; although their analysis is done from MIDI data, they do adopt beat-synchronous observation vectors.

Perhaps the approach which is most similar to ours is that proposed by [SE03a] for chord estimation. In this approach an HMM is used on Pitch Class Profile features (PCP) estimated from audio. Both the models for chords (147 of them) and for chord transitions, are learned from random initializations using the expectation maximization (EM) algorithm. Importantly, this approach differs from ours on that no musical knowledge is explicitly encoded into the model, something that, as will be demonstrated in future sections, has a notable impact on the robustness of the estimation. Also, our choice of feature set and use of a beat-synchronous analysis frame minimizes the effect of local variations. Finally, our proposal differs in scope, we are not trying to achieve chord transcription but to generate a robust harmonic blueprint from audio, and to this end we limit our chord lexicon to the 24 major and minor triads, a symbolic alphabet that we consider to be sufficient for similarity-based applications.

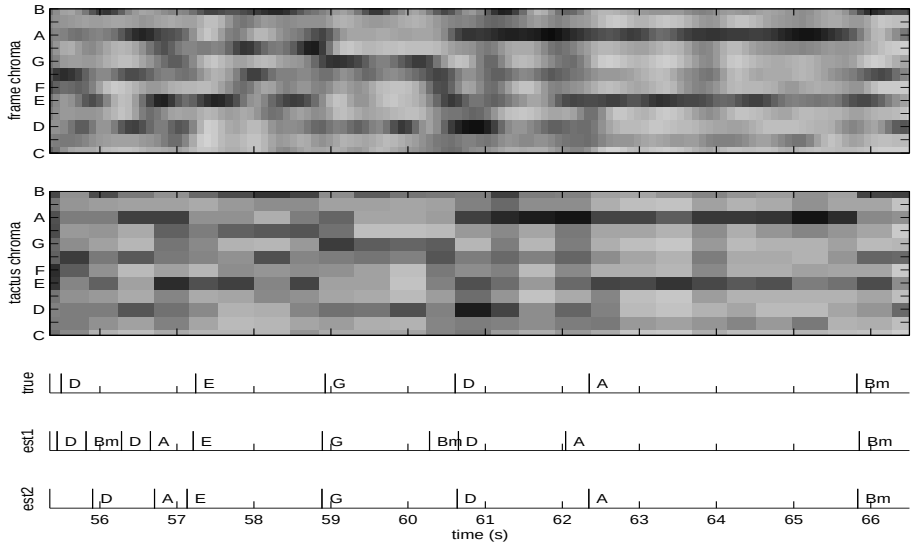


Figure 3.17: Frame and tactus-based feature vectors for *Eight days a week* by The Beatles. At the bottom the estimated chord labels can be observed: "true" corresponds to the ground-truth chord annotation, "est1" corresponds to the chord labels estimated using frame-based features, and "est2" corresponds to the chords estimated using tactus-based features.

3.3.2 Feature set

The first stage of our analysis is the calculation of a sequence of suitable feature vectors. The process can be divided into three main steps: 36-bin chromagram calculation and tuning, beat-synchronous (tactus) segmentation and 12-bin chromagram reduction.

Chromagram calculation and processing

As in the previous section, we use the constant Q transform [[Bro91a]] for the generation of a 36-bin chromagram. We also replicate the principle of chromagram tuning, but using a simpler version of the tuning algorithm proposed by [HS05] and explained in the previous section. The algorithm starts by picking all peaks in the chromagram. Resulting peak positions are quadratically interpolated and mapped to the $[1.5, 3.5]$ range. A histogram is generated with this data, such that skewness in the distribution is indicative of a particular tuning. A corrective factor is calculated from the distribution and applied to the chromagram by means of a circular shift. Finally, the tuned chromagram is low-pass filtered to eliminate sharp edges.

Beat-synchronous segmentation

As mentioned before, beat-synchronous analysis of the signal helps to overcome the problems caused by transient components in the sound, e.g. drums and

guitar strumming, and short ornamentations, often introduced by vocals. Both these cases are quite common in pop music recordings, hence the relevance of this processing step. Furthermore, harmonic changes often occur at a longer time span than that defined by the constant Q analysis, thus the default temporal resolution results unnecessary and often detrimental.

In our approach we use the beat tracking algorithm proposed by [DP05b] and presented in Section 2.3. This method has proven successful for a wide variety of signals. Using beat-synchronous segments has the added advantage that the resulting representation is a function of beat, or "tactus", rather than time. These facilitates comparison with songs in different tempos.

Observation Vectors

Finally, the chromagram is averaged within beat segments and further reduced from 36 to 12 bins by simply summing within semitones. A piece of music is thus represented as a sequence of these 12 dimensional vectors.

3.3.3 Chord Labeling using HMM

Let us turn our attention to the chord labelling of the chroma sequence. Recall, however, that our goal is not true harmonic analysis, but a mid-level representation which we believe will be useful for music similarity and music retrieval tasks. For this we apply the HMM framework [Rab89]. As mentioned in section 3.3.1, we are not the first to use this framework, but we utilise it in a relatively new way, based largely on music theoretic considerations.

Chord lexicon

The first step in labeling the observations in a data stream is to establish the lexicon of labels that will be used. We define a *lexical chord* as a pitch template. Of the 12 octave-equivalent (mod 12) pitches in the Western canon, we select some n-sized subset of those, call the subset a *chord*, give that chord a name, and add it to the lexicon. Not all possible chords belong in a lexicon and we must therefore restrict ourselves to a musically-sensible subset. The chord lexicon used in this work is the set of 24 major and minor triads, one each for all 12 members of the chromatic scale: C Major, c minor, C♯ Major, c♯ minor ... B♭ Major, b♭ minor, B Major, b minor. Assuming octave-invariance, the three members of a major triad have the relative semitone values n, $n+4$ and $n+7$; those of a minor triad n, $n+3$ and $n+7$. No distinction is made between enharmonic equivalents (C♯/D♭, A♯/B♭, etc.).

We have chosen a rather narrow space of chords. We did not include dyads nor other more complex chords such as augmented, diminished, 7^{th} or 9^{th} chords. Our intuition is that by including too many chords, both complex and simple, we run the risk of "overfitting" our models to a particular piece of music. As a quick thought experiment, imagine if the set of chords were simply the entire $\sum_{n=1..12} \binom{12}{n} = 2^{12} - 1$ possible combinations of 12 notes. Then the set of chord labels would be equivalent to the set of 12-bin chroma and one would not gain any insight into the harmonic "substance" of a piece, as each observation would likely be labelled with itself. This is an extreme example but it illustrates the intuition that the richer the lexical chord set becomes, the more our feature

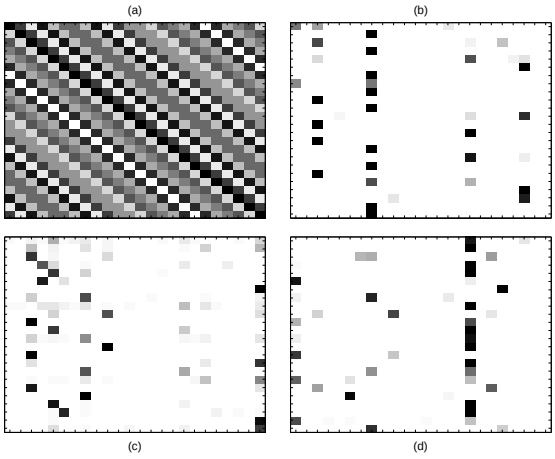


Figure 3.18: State-transition distribution A: (a) initialisation of A using the circle of fifths, (b) trained on *Another Crossroads* (Michael Chapman), (c) trained on *Eight days a week* (The Beatles), and (d) trained on *Love me do* (The Beatles)

selection algorithms might overfit one piece of music and not be useful for the task of determining music similarity.

While it is clear that the harmony of only the crudest music can be reduced to a mere succession of major and minor triads, as this choice of lexicon might be thought to assume, we believe that this is a sound basis for a probabilistic approach to labelling. In other words, the lexicon is a robust mid-level representation of the salient harmonic characteristics of many types of music, notably popular music.

HMM initialization

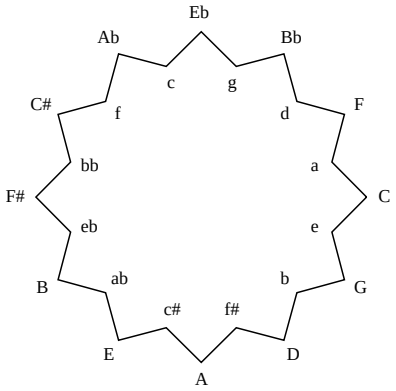
We are not going to cover the basics of hidden Markov modelling. This is far better covered in works such as [Rab89] and even by previous music HMM works cited above. Instead, we begin by describing the initialisation procedure for the model. As labelled training data is difficult to come by, we forgo supervised learning and instead use the unsupervised mechanics of HMMs for parameter estimation. However, with unsupervised training it is crucial that one start the model off in a reasonable state, so that the patterns it learns correspond with the states over which one is trying to do inference.

Initial state distribution [π]

Our estimate of π is $\frac{1}{24}$ for each of the 24 states in the model. We have no reason to prefer, a priori, any state above any other.

State transition matrix [A]

Prior to observing an actual piece of music we also do not know what states are more likely to follow other states. However, this is where a bit of musical knowledge is useful. In a song, we might not yet know whether a C major triad is more often followed by a $B^{\flat}$ major or a D major. But it is reasonable to assume that both hypotheses are more likely than an $F^{\sharp}$ major. Most music tends not to make large, quick harmonic shifts. One might gradually wander from the C to the $F^{\sharp}$, but not immediately. We use this notion to initialise our state transition matrix.



The figure above is a doubly-nested circle of fifths, with the minor triads (lower case) staggered throughout the major triads (upper case). Triads closer to each other on the circle are more consonant, and thus receive higher initial transition probability mass than triads further away. Specifically, the transition C→C is given a probability $\frac{12}{144}$, C→e = $\frac{11}{144}$, C→G = $\frac{10}{144}$, and so on in a decreasing manner, until C→$F^{\sharp}$ = $\frac{0}{144}$. At that point, the probabilities begin increasing again, with C→$b^{\flat}$ = $\frac{1}{144}$ and C→a = $\frac{11}{144}$.

The entire 24×24 transition matrix, as seen in Figure 3.18(a), is constructed in a similar manner for every state, with a state's transition to itself receiving the highest initial probability estimate, and the remaining transitions receiving probability mass relative to their distance around the 24-element circle above.

Observation (output) distribution [B]

Each state in the model generates, with some probability, an observation vector. We assume a continuous observation distribution function modelled using a single multivariate Gaussian for each state, each with mean vector μ and covariance matrix Σ. [SE03a] use random initialisation of μ and a Σ covariance matrix with all off diagonal elements set to 0, reflecting their assumption of completely uncorrelated features. We wish to avoid this assumption. One of the main purposes of this work is to argue that musical knowledge needs to play an important role in music information retrieval tasks. Thus if we are using triads as our hidden state labels, μ and Σ should reflect this fact.

Let us take for example the C major triad state. Instead of initialising μ randomly, we initialise it to 1.0 in the C, E, and G dimensions, and 0.0 elsewhere. This reflects the fact that the triad is grounded in those dimensions. Initialisations of μ for all states can be seen in Fig. 3.19(a).

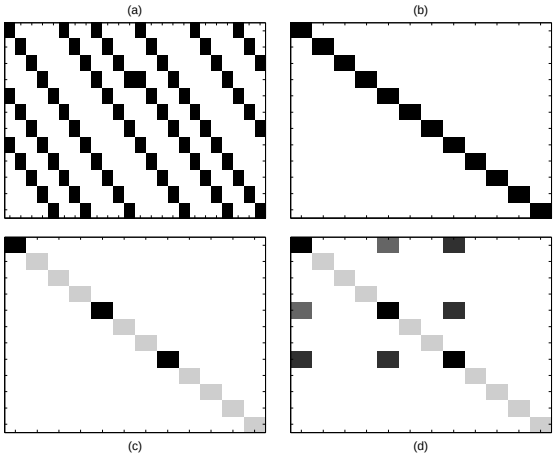


Figure 3.19: Initialisations for μ and Σ. Top-left is μ for all states (a). Then for a C major chord: diag-only (b), weighted-diag (c), and off-diag(d) initialisations of Σ.

The covariance matrix should also reflect our musical knowledge. Covariance is a measure of the extent to which two variables move up or down together. Thus, for a C major triad, it is reasonable that pitches which comprise the triad are more correlated than pitches which do not belong to the triad. Naturally, the pitches C, E, and G are strongly correlated with themselves. Furthermore, these pitches are also strongly correlated with each other. We symmetrically use the knowledge, gained both from music theory as well as empirical evidence [Kru90], that the dominant is more important than the mediant in characterising the root of the triad. We set the covariance of the tonic with the dominant to 0.8, the mediant with the dominant to 0.8, and the tonic with the mediant to 0.6. The actual values are heuristic, but the principle we use to set them is not.

The remainder of covariances in the matrix are set to zero, reflecting the fact that from the perspective of a C major triad there is little useful correlation between, say, an $F^{\sharp}$ and an $A^{\sharp}$. The non-triad member diagonals are set to 0.2 both to indicate that non-triad pitches need not be as strongly self-correlated, as well as to insure that the matrix is positive, semi-definite. Figure 3.19(d) shows the covariance matrix used for the C major triad state.

The covariance for C minor is constructed almost exactly the same way, but with the mediant on $D^{\sharp}/E^{\flat}$ rather than on E, as would be expected. The remainder of the matrices for all the states are constructed by circularly shifting the major/minor matrix by the appropriate number of semitones.

HMM Training

A key difference between our approach and previous systems is our use of musical knowledge for model initialization. There are two important pieces of information that we are providing the system: a template for every chord in the lexicon, as given by μ and Σ, and cognitive-based knowledge about likely chord

progressions, as given by the state transition probability matrix A.

It is relatively safe to say that the template for a chord is almost universal, e.g. a C major triad is always supposed to have the notes C, E and G. If we were to change our chord models from song to song we cannot longer assume that a certain state will always map to the same major or minor triad. Our labels would not have universal value. Furthermore, it is very unlikely that all chords in our lexicon will be present in any given song (or on any reasonably sized training set), and in training, this situation gives rise to the undesirable effect of different instances of existing chords being mapped to different (available) states, usually those that are initialized closely, e.g. relative and parallel minors and majors.

On the other hand, chord progressions are not universal, changing from song to song depending on style, composer, etc. Our initial state transition probability matrix provides a reference, founded in music cognition and theory, on how certain chord transitions are likely to occur in most western tonal music, especially pop music. We believe that this knowledge captures the *a-priori* harmonic intuition of a human listener. However, we want to provide the system with the adaptability to develop models for the particular chord progression of a given piece (see Fig. 3.18), much as people do when exposed to a piece of music they have never heard before.

We therefore propose selectively training our model using the standard expectation maximization (EM) algorithm for HMM parameter estimation [[Rab89]], such that we disallow adjustment of $B = \{\mu, \Sigma\}$, while π and A are updated as normal. We believe this kind of selective training to provide a good trade-off between the need for a stable reference for chords, and a flexible, yet principled, modeling of chord sequences.

Chord Labeling (Inference)

Once we have both a trained model and an observation sequence, we can apply standard inference techniques [[Rab89]] to label the observations with chords from our lexicon. The idea is that there are many sequences of hidden states that could have been responsible for generating the chroma vector observation sequence.

The goal is to find that sequence that maximizes the likelihood of the data without having to enumerate the exponentially many (24^n, for a sequence of length n, in our model) number of sequences. To this end a dynamic programming algorithm known as Viterbi is used. This algorithm is well covered in the literature and we do not add any details here.

3.3.4 Evaluation and Analysis

In summary, our system, for a single piece of music, is:

1. Compute the 36-bin chromagram for the music piece.
2. Tune the chromagrams (globally) to remove slight sharpness or flatness and avoid energy leaking from one pitch class into another

3. Segment the signal frames into tactus-sized windows, average the chroma within each window, and finally reduce each chroma from 36 to 12 bins by summing all three bins for each pitch class

4. Selectively train the HMM to get a sense of the harmonic movement of the piece

5. Decode the HMM (do inference) to give a good mid-level harmonic characterization of the piece

Despite our stated goal of harmonic description rather than analysis, we found that it is still useful to attempt quantitative evaluation of the goodness of our representation by comparing the generated labels to an annotated collection of music. We use the test set proposed and annotated by [HS05], following the guidelines in Appendix B. It contains 28 recordings (mono, $f_s = 44.1kHz$) from the Beatles albums: *Please Please Me* (CD1) and *Beatles for Sale* (CD2). Note that all recordings are polyphonic and multi-instrumental containing drums and (multi-part) vocals.

The majority of chords (89.51%) in the manually labeled test set belong to our proposed lexicon of major and minor triads. However, the set also contains more complex chords such as major and minor 6^{ths}, 7^{ths} and 9^{ths}. For simplicity, we map any complex chord to its root triad, so for example C#m7sus4 becomes simply C#m. If anything, this mapping has the effect of overly penalizing our results, as chords of 4 or more notes could contain triads other than its root triad, e.g. Fm7 (F, G#, C, D#) has 100% overlap with G# (G#, C, D#) and Fm (F, G#, C). Comparisons are made on a frame-by-frame basis, such that a true positive is defined as a one-to-one match between estimation and annotation.

To quantitatively demonstrate some of the hypotheses put forward on this paper, we evaluate a series of incremental improvements to our approach. Figure 3.4 shows the model parameters for each experiment and its corresponding results for the test set (in percentage of true positives). Results are presented per CD and in total. The considered model parameters are:

- Feature scope: Whether it is a frame-by-frame (time-based) or a beat-synchronous (tactus-based) chroma feature set.

- Initialization of A: Whether it is randomly initialized or initialized according to the circle of fifths.

- Initialization of B: Whether Σ is initialized as a diagonal matrix with elements equal to 1.0 (diag-only, Fig. 3.19(b)), whether it is the diagonal with weighted triad elements, as in Fig. 3.19(c), and off-diagonal elements set to 0.0 (weighted-diag), or whether it includes the mediant and dominant off-diagonal elements, i.e. the Fig. 3.19(d) matrix (off-diag).

- Training: Whether π, A and B are updated in the expectation-maximization step of HMM training or whether B is left fixed and only π and A are adjusted.

Results in Figure 3.4 clearly support the choices made in this paper. The first three rows show how initializing Σ with a weighted diagonal and off-diagonal elements outperforms diagonal-only initializations. This supports the view that the

feature set is highly correlated along the dimensions of the elements of a chord. The weighted diagonal in itself introduces a noticeable amount of improvement over the unitary diagonal, a further indication of the strong correlation between the tonic, mediant and dominant of a chord.

The initialization of A using the circle of fifths brings about more than 10% relative improvement when compared to the random initialization. This shows how the use of musical knowledge is crucial. .

From the analysis of the last two rows in Figure 3.4 two more observations can be made. The first is that selective training introduces considerable benefits into our approach. The huge accuracy increase (from 42.93% to 75.04%) supports the view that the knowledge about chords encoded in B is universal, and as such it should not be modified during training. This accuracy increase occurs for every song, showing the generality of this assertion.

The second observation is that the use of a tactus-based feature set clearly outperforms the frame-by-frame estimation. This point is further illustrated by the chord estimation example in Fig. 3.17, where the frame-by-frame estimation is subject to small variations due to phrasing or ornamentation (as shown by the spurious estimations of B minor chords between 56 and 60.5 seconds), while the tactus-based estimation shows more stability and, therefore, accuracy when compared to the ground-truth annotation. Furthermore, chord changes are more likely to occur on the beat, thus chords detected using the tactus-based feature set tend also to be better localized.

Our results compare favorably to those reported by [SE03a] and [HS05]. The maximum true positives rate in the collection is 90.86% for *Eight days a week*. Conversely, the worst estimation is for *Love me do*, with only 49.27% of chords correctly identified. For the latter case almost all errors are due to relative minor confusions: C being confused with E minor consistently through the song. As we will see in the next section the consistency of the representation, even when wrong, can be useful for certain applications.

3.3.5 Discussion

The main contribution of this work is the creation of an effective mid-level representation for music audio signals. We have shown that by considering the inherent musicality of audio signals one achieves results far greater than raw signal processing and machine learning techniques alone (Figure 3.4). Our hope is that these ideas and their results will encourage those in the field working on raw audio to build more musicality into their techniques. At the same time, we hope it also encourages those working on the symbolic side of music retrieval to aide in the creation of additional musically sensible mid-level representations without undue concern over whether such representations strictly adhere to formal music theory guidelines.

In support of this goal, we have integrated into a single framework a number of state-of-the-art music processing algorithms. Specifically, we build our algorithms upon a musical foundation in the following ways: (1) The audio signal is segmented into tactus windows rather than time-based frames. (2) Pitch chroma are tuned. (3) A lexicon of 24 triads is used, which is neither too specific or too general, in an attempt to *describe* harmonic movement in a piece rather than doing a formal harmonic analysis. (4) Initialisation of the machine learning (HMM) algorithm is done in a manner that respects the dependency

between tonic, mediant, and dominant pitches in a triad, as well as the consonance between neighbouring triads in a sequence. Finally, (5) the machine learning algorithm itself is modified with an eye toward musicality; updates to model parameters are done so as to maintain the relationship between pitches in a chord, but be amenable to changing chord transitions in a sequence.

In the future we are planning a series of audio-to-audio music retrieval experiments to further show the validity of our approach. We will also continue to develop and integrate techniques that emphasise the musical nature of the underlying source. We believe that this mindset is vital to continuing development in the field.

3.4 New research: Object-oriented analysis of pitched musical audio signals

In the context of auditory scene analysis [Bre90] we can consider the musical contents of a signal as objects. An object can be roughly defined as the signal form of a note or a monophonic note sequence. For high-level semantic description, the ultimate goal is to deal with objects that are attached to significant musical constructs such as chords, phrases or sections. However for the preliminary work in this section, the word "object" refers mostly to elemental tonal events in the audio signal.

The objective of object-oriented sound analysis is to bridge the gap between the raw audio signal and the "objectified" audio signal. In this section, the task is restricted to pitched sounds, which cover most tonal musical instruments. Raw single-channel audio signals are used as input. A set of object-oriented descriptors based on harmonic sinusoidal modelling are then retrieved from the raw signal to describe objects.

For a single note, its corresponding descriptor tries to capture the following properties: an interval in time where the note exists; frequencies of its k^{th} partial for some positive k's at certain sampling instants; amplitudes and phase angles of its k^{th} partial for some positive k's at certain sampling instants.

In theory these descriptors are nearly enough for re-synthesizing the note sound (as demonstrated by methodologies such as sinusoidal modelling synthesis - SMS). However, in practice the re-synthesized note will differ from its original recording, unless the original sound is already monophonic. This is either due to practical computational problems with real-world recordings, or due to perceptual phenomena of human hearing.

On the other hand, by capturing some of the most important sound parameters, these descriptors serve to provide high-level information about the signal contents. As a result they can be widely used for interfacing note-level operations. Related applications may include:

- Note-level audio editing: or changing audio parameters of a note without affecting other parts of a sound. You may extract a note from the original audio by filtering at its partial frequencies or by synthesizing from sinusoidal parameters. You may suppress a note in the original audio by notching at its partial frequencies or by subtracting its sinusoidal partials from the original sound. You may change its volume by amplifying the partials, and therefore accomplish remixing from already mixed music.

You may make it smooth or rough or percussive by adjusting its amplitude envelope. You may apply sinusoidal-synthesis-based digital audio effects. You may even correct some type of mistakes one made during the performance. These operations are almost unimaginable with raw audio signal, since the related audio parameters are encrypted there in a hard-to-grasp loose manner. But with the mid-level descriptors, they've been made explicit and compact.

- Sound source identification: for music signals this mostly refers to instrument and singer identification. Current musical sound source ID has reached acceptable performance with some clean sound sources, but is heavily troubled when there is musical background. The object-oriented descriptor in this case serves to "tune" the receiver to the partials of a pitched sound within a mixture of multiple sounds, therefore passing on a much cleaner signal to the classifier. This in return introduces another application: annotating a note by locating its presence both in time and in frequency. This task is hardly achievable by a human annotator considering the huge amount of information to be labelled. Such semi-automatic annotation is not guaranteed to be flawless, but helpful for developing MIR systems.

- Automatic music transcription: since important note-level information is mostly preserved in the new descriptor, it's possible to start music transcription from it rather than from raw audio data. As the descriptors are based on sinusoidal modelling, then it is convenient to estimate pitch from them. The partial frequency trajectory ($d2$) encloses a history of pitch change during a single event. For instruments whose partial frequencies remain constant, such as oboe and piano, the fundamental frequency trajectory can be smoothed to yield the pitch. For those whose partial frequencies vary within a note, such as violin and human voice, the trajectory itself is a more accurate measure than any single pitch. However, to derive a single pitch from the swinging fundamental frequency is not a trivial job.

- Performance-related parameter estimation: Interests on studying musical performance using signal processing techniques have been developing for both vocal and instrumental music. Unlike music transcription which aims at deriving pitches on a pre-defined musical note scale, performance studies require more detailed information on the events that give music its flavour. Related "expressive music" parameters include tempo and pitch variation, amplitude and frequency modulation, volume contrast, etc., which can be estimated from the object-oriented descriptors.

3.4.1 Related techniques

There are few reported techniques for extracting the proposed descriptors from a sound mixture. However, some existing works are highly related to the proposal, as discussed below.

Sinusoidal modelling

This models any sound as the additive mixture of time-varying sinusoids. These sinusoids make little semantic sense themselves. The model does not distinguish components of one event from those of another, so is less capable of generating note-level information [MQ86, Ser89]. Methods used in sinusoidal modelling, such as sinusoid measurement techniques, are useful for developing the proposed descriptors, which models the sound as the mixture of harmonic near-sinusoids in some background.

Auditory scene analysis

This tries to achieve a goal similar to the proposed descriptors by a breaking-grouping scheme. In breaking stage the audio signal is decomposed into time-frequency particles from which the group stage composes meaningful sound objects using auditory cues such as spectral continuity and common onsets/offset/modulations [BC94, Ell96]. The two systematic reports on this research area have given only some primitive implementation and results. Recent advancements are rarely reported. In my work a different approach will be taken: rather than breaking and grouping, a partial searching scheme is used to generate event trajectories in time-frequency plane.

Iterative subtraction

This tries to resolve multiple pitches from a single spectrum by an iterative detection-subtraction loop. In each iteration a pitch is detected and then partials corresponding to the pitch are subtracted from the original signal [Kla03a]. To derive the proposed descriptors however, subtraction seems to be too destructive to the residue background. We propose using an iterative addition scheme instead, e.g. the $(k+1)^{th}$ component is searched in the complete sound signal referencing the previous k concurrent components.

Bayesian harmonic model

This tries to resolve all harmonic components in the sound using a Bayesian inference method. The model is soluble using Monte Carlo method. Primitive results show it is capable to separate a limited number of sparse signals [DG03].

Methods for source separation using multiple channels, especially 2 channels, have been reported successful with limited number of sound source (as with BSS) or with well-controlled recording conditions (as with DUET - see Chapter 4). For mono recordings, dictionary-based methods have also been proposed with some success for specific recordings.

Methods for generating from music recordings note-level descriptors, described at the beginning, are however less heard of. Nor are the proposed applications such as the intelligent note-level audio editing or the object-focused instrument identification.

3.4.2 Methodology

We propose an additional layer between audio data and the proposed descriptors, which makes no hard decision on what is there and what it is like, but

only evaluates how likely something is there, and how much it contributes to the whole thing. Briefly speaking, it looks for an event by first hypothesizing its existence and then evaluating it. A complete hypothesis of an event is consists of descriptors $d1$ and $d2$, i.e. its partial frequency trajectories, or energy support, in the time-frequency plane. Similarly, it looks for two concurrent events by first hypothesizing their partial frequency trajectories and than evaluating their combined contribution. This differs from the iterative subtracting method in that the existence of a first object does not corrupt the detection of a second, even if they share partials. As long as there is a hypothesis pool from which hypotheses are drawn, the hypothesis that's evaluated to contribute the most can be picked as the result of descriptors $d1$ and $d2$.

The hypothesis pool, or the allowable set of hypotheses, is determined by some pre-defined criteria, including harmonicity assumption and frequency continuity assumption. However, even with these constraints, the pool is too huge a set to exhaust. So rather than composing a dictionary of hypotheses in advance, we generate the hypothesis pool from the data. Like in sinusoidal modelling, we start from spectral peaks — since any event in an additive mixture have to involve some local spectral maxima to make audible contribution. With any spectral peak at time t and frequency f, a set of "incomplete" hypotheses can be derived, putting the point on the 1^{st}, 2^{nd}, ..., etc. partial respectively. An incomplete hypothesis is a starting point to grow into a complete hypothesis by grabbing in more spectral peaks to fill the vacancies. The growth of a hypothesis involves a horizontal growth along time axis, and a vertical growth with respect to partial index. The harmonicity assumption restrains the position of a vertically adjacent partial within a small frequency interval from which peaks may be selected to grow the incomplete hypothesis. Similarly the frequency continuity assumption, on the other hand, restrains the position of a partial in the next moment within a small frequency interval centred at its current position. A partial frequency consistency assumption, which requires all partial frequencies of the same event increase or decrease together, helps to further cut down the number of hypotheses. In case no spectral peak can be located for a hypothesis vacancy that meets all the constraints, an artificial non-peak particle, the place holder, is used instead. The number of event hypotheses generated from a spectral peak in this way can already be enormous and expensive to evaluate, not to mention those of multiple events. To tackle the single-event hypothesis evaluation problem, a dynamic programming method is developed, which allows for 1^{st}-order dependence of partial frequency intervals on previous frequency, and for 2^{nd}-order dependence of partial frequency intervals on lower partials. Given an interval in time and a start point (t,f), the method searches for a hypothesis lying in the interval and containing the point, which maximizes the contribution.

To find multiple concurrent events, one can either simply do the single event growing multiple times and then treat their interactions, or search for the next event while referring to current hypothesis. The first way requires less computation, but tends to lose a spectral peak while allocating a nearby one to multiple events. To avoid making too-early decisions, a tree-searching scheme is implemented that allows multiple candidates at each iteration. The method does not resolve a shared partial into parts to be associated to different events, but may tag some points on a partial trajectory as "shared". [WS05] gives a simple example of how one plays with shared partials to retrieve meaningful

information.

The analyzer operates in spectral domain. Spectral peaks are located using a frequency-domain matched filter derived from sinusoidal modelling, in which the discrete amplitude spectrum is convolved with an analogue window function. Spectral peak frequency is located at the maximum of the convolution, and the corresponding maximal value is the peak amplitude. A faster alternative is using quadratic interpolation, which however does not guarantee local maximality. The phase angle of a spectral peak can be interpolated from local discrete spectrum.

Applying the descriptors for audio editing is mostly a human interface design issue – how the algorithm works greatly depends on what command it receives from the operator. The event-level or note-level audio editing involves two stages: finding the event and editing it. In the first stage human operator interacts with computer to accomplish the task of generating the right descriptors. In the most automatic case, i.e. the computer generates a list of correct events, the human operator only needs to choose which note to work on. Yet involving some human participation in the event finding process also helps, especially for complex signals with which the growth of an event tends to go astray. The second stage is a basic signal processing one. Multiple ways exist to extract an event out using the descriptors. One may use the standard SMS technique as described in [MQ86], frequency domain synthesis with FFT and overlap-add, direct block-wise filtering with overlap-add, or time-varying filtering using known partial frequencies. To suppress an event, one may subtract the SM-Synthesized event signal directly, cancel it out or notch it out in frequency domain with FFT and overlap-add, use direct block-wise notch filtering with overlap-add, or use time-varying notch filtering. However, since the descriptors do not resolve shared partials, event suppression may corrupt what's in the residue. This can be partially eased by referring to descriptors on concurrent events when removing an event.

Applying the descriptors for source identification is a three-stage process. Firstly annotations should be made in the very format of descriptors d1 and d2. This is actually a semiautomatic human-computer interaction process, in which event descriptors are generated by the computer under some annotator instructions and fed back to the annotator to decide whether or what to label. For example the annotator may specify an interval and a point, from which the computer derives an event in the format of d1 and d2, and presents it back probably as an audio clip; the annotation then listen to it to decide whether it's a piano or violin sound. In the second stage sound source models are trained using features derived from the descriptors, this time including d3. In the last stage, i.e. identification test, events are *automatically* generated and submitted to the classifier. In both training and testing stages special care should be taken with shared partials, e.g. by ignoring or smoothing them.

Applying the descriptors for automatic transcription can be straightforward if the events have well-defined fundamentals, such as with piano or oboe, or a fuzzy thing if the events have unstable fundamentals, such as with human singing. Often we may take the local average of the fundamental frequency then choose a closest pitch on the scale. However, no systematic theory guarantees it to be the perceived pitch.

3.4.3 Preliminary results

An algorithm is developed to evaluate how much partials related to a given set of pitches contribute to a spectrum and to grow a set of descriptors from a given starting point in time and frequency.

With Glenn Gould's recording of Bach's Prelude in C BWV 846a the algorithm is able to capture no less than the 10 lowest partials most of the time, which host more than 99.5% of the total energy, with some old-plus-new modelling discussed in [WS05]. This enables some intelligent editing operations such as adjusting the strength of individual notes or changing their damping rates. The descriptors have also been applied for polyphonic transcription. With Glenn Gould's recording of Bach's Fugue in C BWV 846b, a note detection rate of 79.9% is achieved with 30.6% note insertion rate.

3.5 Conclusions

In this chapter we have presented the harmonic component of our semantic description scheme. We have shown our contributions towards the state of the art in tonality estimation, chord identification and harmonic representation for polyphonic multi-instrumental signals. Also, we have introduced some preliminary work on objectifying tonal components in audio signals, a research trend that we expect will traduce on new approaches to the semantic description of harmonic and, more significantly, melodic content from raw audio. Finally, our contributions to the standardisation of chord annotations and the use of audio-alignment approaches to maximise the use of existing annotated data can be seen in Appendices B and C respectively.

Our overall approach to harmonic description, is based on the use of chromagrams as our low-level feature set. Chromagrams relate to the human perception of pitch, and as such were considered as adequate for our task. Different approaches were presented for the estimation and processing of chromagrams.

Initially we have shown how these features can be used for the task of key estimation, by combining the information extracted from the signal with knowledge from studies in music cognition. Then a model based on simple chord templates is proposed for the task of chord estimation, showing how assertive processing of the feature vectors can be used to enhance our identification of chords. Particularly interesting is the use of a tuning procedure to compensate for detuning in the original signal. Finally, we enhance our chord estimation process by using an HMM that embeds music theoretic and cognitive information about the nature of chords and of likely chord progressions. This system also illustrates the combined used of high-level rhythmic and harmonic features, by using a tactus-based analysis window for the chord estimation. We demonstrate the success of these implementations on databases of annotated polyphonic and multi-instrumental music.

Our proposals for harmonic description are currently under extensive study for research associated to Work Packages 2 and 3. In Chapter 6 we will briefly illustrate how harmonic description can be used for long-term segmentation of music signals, and in Chapter 8 we show the use of chromagrams and chords for the task of characterising music similarity between songs in a collection, and for visualising performance attributes.

Test results for:		Correct	RM	Rm	PM	Pm	D	SD	Other
Individual Song:	Every Little Thing	72.9	0	3.1	0.8	0	4.9	0.9	17.4
	Eight Days a Week	77.9	0.1	2.9	0.4	0	5.8	1.2	11.7
Whole Album:	Please Please Me	53.9	1.7	3.1	2.2	0	6.3	1.8	31
	Beatles for Sale	70.8	0.6	2.3	0.8	0	4.4	2.5	18.6
All 28 songs		62.4	1.1	2.7	1.5	0	5.4	2.1	24.8

Errors: **RM** Relative Major **PM** Parallel Major **D** Dominant
Rm Relative Minor **Pm** Parallel Minor **SD** Subdominant

Table 3.3: Frame accuracy percentages and common error percentages from tests.

Parameters						TP %		
Feature scope	π	A	B μ	B Σ	Training	CD1	CD2	TOTAL
tactus	$\frac{1}{24}$	random	template	diag-only	π, A, B	22.88	29.83	26.36
tactus	$\frac{1}{24}$	random	template	weighted-diag	π, A, B	34.14	36.24	35.19
tactus	$\frac{1}{24}$	random	template	off-diag	π, A, B	33.13	44.36	38.74
tactus	$\frac{1}{24}$	circle of 5^{ths}	template	off-diag	π, A, B	38.09	47.75	42.93
frame	$\frac{1}{24}$	circle of 5^{ths}	template	off-diag	π, A	58.96	74.78	66.87
tactus	$\frac{1}{24}$	circle of 5^{ths}	template	off-diag	π, A	**68.55**	**81.54**	**75.04**

Table 3.4: Results for various model parameters

Chapter 4

Describing Timbre

Another dimension of musical description is that defined by the timbre or instrumentation of a song. Extracting truly timbral information from music, as pertaining to separate instruments or types of instrumentation, is an issue that exceeds in difficulty some of the automatic description approaches presented in previous chapters. This is due to the fact that it implies classifying, characterizing and describing information which is buried behind many layers of correlated data, as harmonic and rhythmic information from different instruments in a polyphonic recording is usually musically coherent. Advances have been made for mono-instrumental music, as will be shown in this chapter, but extensions to multi-instrumental ensembles, a category where most commercial music falls, are still to be fully developed. As a consequence, previous related work has concentrated on the characterization of the "overall" timbre or "texture" of a piece of music as a function of sets of low-level features, as will be seen in Chapter 6 for segmentation and in Chapter 8 for similarity.

While some success has been achieved on the above mentioned applications, these *textural* descriptions are usually associated to acoustic features in the audio signal, and often lack meaning in terms of the instrumentation that originated the signal in the first place. These studies are the legacy of research on audio fingerprinting, which is not interested on musical or generative abstractions, but in finding an unique identifier that can distinctively describe a given recording. This is not enough for the approach adopted by the SIMAC project, where we are effectively interested on describing instrumentation.

In this chapter we will start by presenting SIMAC's contribution to the current state of the art in instrument identification of mono-instrumental music (Section 4.1). Then, we will present our novel proposals for the recognition of instrumental sounds and the extraction of high-level timbral descriptors from multi-instrumental recordings. Both these tasks will be constrained to the subspace of percussive sounds and descriptors. Finally, we present work in progress relative to the task of blind source separation as related to the issue of instrumental description.

4.1 Instrument Identification using LSF and K-means

As previously mentioned, automatic instrument recognition systems provide essential extra capabilities for Music Information Retrieval (MIR) and high level musical content analysis applications. Retrieving pieces of music containing a given instrument, looking for similarities in different musical extracts or classifying pop songs in accordance with the lead singer voice are possible examples of user end applications.

In this section we describe a mono-feature system using the Line Spectrum Frequencies as features and a K-means algorithm as classifier. More details can be found in [CDS05a]. We only recall here the main system principles and the conclusions.

4.1.1 Background

Various attempts have been made to construct automatic musical instrument recognition systems. Different approaches and scopes have been used achieving

different performances. State of the art systems in monophonic instrument recognition use solo recordings of sustained notes or excerpts extracted from *real* solo phrases. Two types of systems can be distinguished:

Multifeatures systems: in this approach, it is assumed that timbre cannot be modelled using one type of feature and models of instruments include a mixture of spectral and temporal descriptors (this is partly due to the relatively different physical mechanisms of sound production between musical instruments). Examples of such approach can be found in [ALP01] and [ERD04].

Monofeature systems: in a monofeature system, one unique type of feature is retained to build the instrument models. Although it is agreed that timbre cannot be efficiently modelled by only one feature, these systems allow a better understanding of the interaction feature/classifier. Examples can be found in [Bro99] and [KS04].

System	**Correct Identification**	**Number of instruments**
Marques [MM99]	70.0%	8 (0.2 seconds test)
	83.0%	8 (2 seconds test)
Brown [Bro99]	94.0%	2
Brown [BHM01]	84.0%	4
Martin [Mar98]	71.6%	14
Agostini [ALP01]	92.8%	27
	95.3%	20

Table 4.1: *Recognition performances for five systems using monotimbral phrases.*

Table 4.1 summarises typical results for five systems. In [Bro99], Brown used speaker recognition techniques for classifying between oboe and saxophone. Using cepstral coefficients based on a constant-Q transform, 94% correct identification was reported. In a later study [BHM01], her system classified between oboe, saxophone, flute and clarinet. The most successful feature set was the frequency derivative of 22 constant-Q coefficients measuring the spectral smoothness. A performance of 84% correct identification was reported using a standard GMM classifier. In [MM99], Marques described a system capable of recognising between 8 different instruments (bagpipes, clarinet, flute, harpsichord, organ, piano, trombone and violin). Using 16 Mel-Frequency-Cepstral-Coefficients (MFCC) and a Support Vector Machine (SVM) as classifier, 70% of correct identification was reported for 0.2 second test samples and 83% for 2 seconds in length audio samples. In [Mar98], Martin used a large set of 31 features including the pitch, spectral centroid, attack asynchrony, ratio of odd-to-even harmonic energy (based on the first six partials) and the strength of vibrato-tremolo calculated from the output of a log-lag correlogram. k-NN classifier was used within a taxonomic hierarchy after having applied a Fischer discriminant analysis ([McL92]) on the feature data set in order to reduce the required number of training samples. For 1023 isolated tones over the full pitch range of 14 instruments, 71.6% correct accuracy for the identification of individual instruments has been reported. Finally, Agostini described in [ALP01] a system using the mean and the standard deviation of 9 features derived from a STFT, including the spectral centroid, spectral bandwidth, harmonic energy percentage, inharmonicity and harmonic energy skewness. The three last parameters

were calculated for the first four partials. The best results have been achieved using a Quadratic Discriminant Analysis (QDA) classifier (92.8% for 27 instruments and the maximum 95.3% for 20 instruments), followed by a SVM (69.7% for 27 instruments), a Canonical Discriminant Analysis (CDA) (66.7% for 27 instruments) and finally a k-NN classifier (65.7% for 27 instruments).

As it can be seen from the figures above, problems arise when one tries to compare the performances of different systems. Considered instruments, their number and the audio samples used for the training and testing are usually different. Implementation of the feature extraction function and/or the classifier may also differ from one system to another. This obviously affects the robustness and the validity of a figure to figure comparison. However, general trends and systems behaviours can be exploited and used for the design of new classification algorithms.

4.1.2 Feature set

The spectral characteristisation of audio signals remains the foundation of the conventional understanding of musical sounds by humans. Spectral descriptors including spectral parameters and spectral envelopes are often included in musical instrument identification systems.

Widely used in speech compression, the LSF are derived from the Linear Prediction Polynomial Coefficients (LPC). To a certain extent, the source-filter model of speech production is transposed here to the physical process of sound creation by musical instruments. Representing at the same time the signal short-term spectral envelope and energy distribution (two close LSF values characterise a peak in the spectrum), it has been shown in [KS04] that the LSF are good candidates for modelling the timbre of a sound.

In a source-filter configuration, the short segment of a signal is assumed to be generated as the output of an all-poles filter $H(z) = 1/A(z)$, where A(z) is the inverse filter given by

$$A(z) = 1 + a_1 z^{-1} + ... + a_p z^{-p}, \tag{4.1}$$

p is the order of the LPC analysis and $\{a_i\}, i = 1, ..., p$ the filter coefficients.

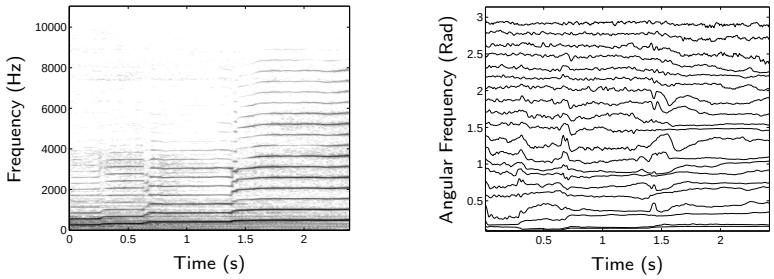
Introduced by Itakura in [Ita75], the Line Spectrum Pairs (LSP) are the roots of two polynomials $P(z)$ and $Q(z)$ defined as

$$\begin{cases} P(z) = A(z) + z^{-p+1}A(z^{-1}) \\ Q(z) = A(z) - z^{-p+1}A(z^{-1}) \end{cases} \tag{4.2}$$

It can be shown that they are interlaced and on the unit circle. Their corresponding angular frequencies are called the Line Spectrum Frequencies. A computationally efficient way to calculate these roots using Chebychev polynomials can be found in [KR86].

4.1.3 Classifier

Musical instruments produce sounds whose timbral properties change with pitch, loudness and playing style. In order to capture the class intra-variability, statistical classifiers are used to learn the statistical distribution of the features that have been previously extracted from a preprocessed set of waveforms. The



(a) Spectrogram (b) LSF

Figure 4.1: Spectrogram (a) and Line Spectrum Frequencies (b) representations for a saxophone solo excerpt with a 20th prediction order. LSP are in the range [-1 1] while the LSF lay in the range [0 π]

k-means is described in this section. In the following, it is assumed that the training data set consists of T observations $X = \{\vec{x}_t\}_{t=1,...,T}$ of LSF vectors.

4.1.4 Instrument modelling

By performing a k-means on the training data set, X is represented by a discrete set of M codevectors. The algorithm is based on the classical LBG implementation described in [LBG80]. Its principle is recalled below.

Given an initial dictionary $C = \{\vec{c}_i\}_{i=1,...,M}$, and the training set X, the two-stages procedure consists of iteratively calculating:

1. **Nearest Neighbours:** Each observation in the data set if affected to the closest codeword in the dictionary. Clusters $R_i, i = 1, ..., M$ are defined by:
$$\begin{cases} R_i = \{\vec{x}_t \in T : d(\vec{x}_t, \vec{c}_i) \leq d(\vec{x}_t, \vec{c}_j); j \neq i\} \\ 1 \leq i, j \leq M \end{cases} \tag{4.3}$$

2. **New codewords:** at each iteration and for each cluster $R_i = \{\vec{x}_k\}_{k=1,...,l_i}$ containing l_i vectors, the new codeword $\vec{c}_i$ is defined as the mean of the data populating the cluster:
$$\vec{c}_i = \frac{1}{l_i} \sum_{k=1}^{l_i} \vec{x}_k, \quad 1 \leq i \leq M \tag{4.4}$$

The choice of a relevant distance d used for the affectation of the data to the nearest neighbours is one of the main algorithm parameter. A commonly used metric is the Euclidean distance defined for two D-dimensional vector $\vec{x}$ and $\vec{y}$ by:
$$d(\vec{x}, \vec{y}) = \sum_{i=1}^{D} (\vec{x}_i - \vec{y}_i)^2 \tag{4.5}$$

In practice, the initial codebook is obtained by the splitting technique. Starting with one codeword (the mean of the all the data set), each vector in the dictionary is iteratively split into two vectors until the closest inferior power of two of M is reached. Finally, the maximally populated cluster is split into two until the desired number of centroids is attained. In our implementation, splitting is performed by adding a small perturbation to the vector coordinates proportional to the standard deviation in each direction. The two-stage procedure described above is then iterated until the average total distortion

$$\overline{d} = \frac{1}{T} \sum_{i=1}^{T} \min_{1 \leq k \leq M} d(\vec{x}_i, \vec{c}_k) \tag{4.6}$$

reaches a local minimum.

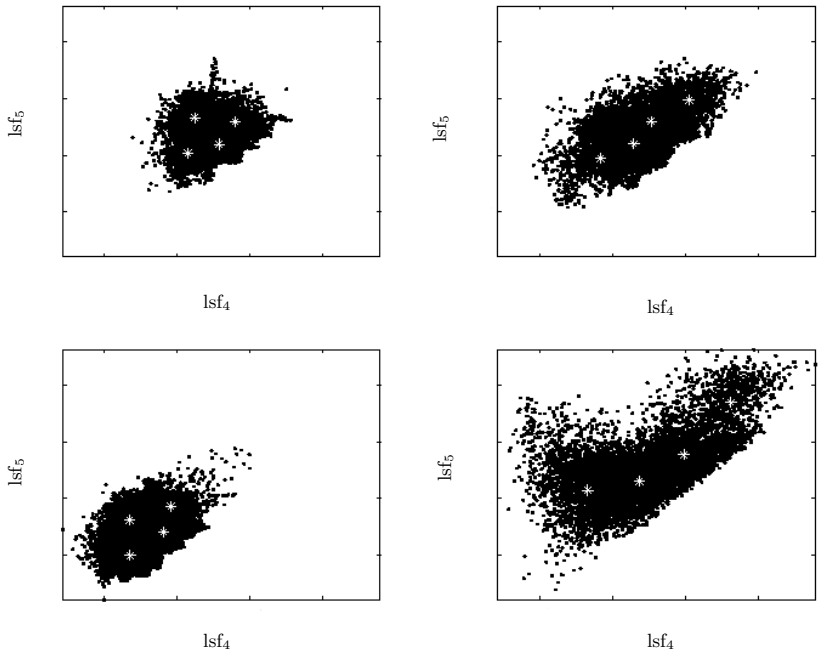


Figure 4.2: Two-dimensional PDF modelling using 4 clusters and a K-means algorithm (b). Training data set corresponds to 2 minutes of monophonic recording. Instruments are (clockwise from top left): accordion, acoustic guitar, piano and spoken speech (male speaker)

4.1.5 Instrument identification

In the following, we will assume that the instrument to identify is represented by a sequence $Y = \{\vec{y}_s\}_{s=1,...,S}$ of LSF vectors. The N instruments in the database are represented by their codebooks $C_n, n = 1, ..., N$ of M codewords. Two minimum distance classifiers between the observation Y and the models are presented in this section.

During the identification phase, the k-means is used to build a codebook $\tilde{C} = \{\vec{\tilde{c}}_m\}_{m=1,\ldots,M}$ which statistically represents the observation Y. The idea beneath is to fully take advantage of the available observation prior to the classification. A distortion measure between two codebooks is introduced in the following. Considering that:

1 Running the k-means algorithm twice on the same data set does not ensure that the codewords are similarly ordered. In other words, it is not guaranteed that

$$\sum_{m=1}^{M} d(\vec{\tilde{c}}_{m,1}, \vec{\tilde{c}}_{m,2}) = 0 \tag{4.7}$$

2 In the case where $\tilde{C}$ contains M times the same codevector identical to one codevector of C, the measure should verify $d_{\tilde{C},C} = 0$,

we propose the following distortion measure between the codebook $\tilde{C}$ and one codebook C_n taken from the database:

$$d_{\tilde{C},C_n} = \sum_{m=1}^{M} \left[\min_{1 \leq m \leq M} d(\vec{\tilde{c}}_m, \vec{c}_{n,m}) \right] \tag{4.8}$$

in which d is the Euclidean distance defined in Eq. 4.5. The distortion measure verifies : $d_{\tilde{C},C} \geqslant 0$ and $d_{C,C} = 0$ but is not symmetric.

Finally, the identity of the unknown codebook is retrieved by finding n^* such that:

$$n^* = \arg \min_{1 \leq n \leq N} d_{\tilde{C},C_n} \tag{4.9}$$

4.1.6 Experiments

The LSF are extracted on signals sampled at 44100 Hz. Silence and low level segments are firstly discarded (only frames having an average energy level above -90 dB are retained) and DC bias are removed using a first-order IIR high-pass filter of frequency response $H(z) = (1 - z^{-1})/(1 - 0.999z^{-1})$. Amplitudes are then normalised to the 0 dB level, and a pre-emphasis ($H(z) = 1 - 0.97z^{-1}$) is used to increase the relative high frequency energy components. This step is particularly useful when LPC-based models are used as it helps the algorithm to better pick the envelope high frequency structure. Next, features are extracted every 17 ms within frames of 23 ms in duration that have been previously weighted by a Hanning window. The LSF are calculated using the algorithm described in [KR86].

The database consists of a combined subset of the IR-IOWA [oI] and the RWC [Dat] isolated note databases. The following instruments have been retained for the experiments: bassoon, clarinet, flute, oboe, sax, trombone, trumpet, cello, violin and viola. A singing voice class containing samples from male and female singers has also been added. In all the experiments, models have been trained using 70% of the available data while 30% were retained for the testing. Experiments were reiterated 3 times with different training and testing sets and identification rates were accordingly averaged.

Parameters are the number of clusters for both GMM and K-means and the prediction order for the LPC analysis. More details about the experiments

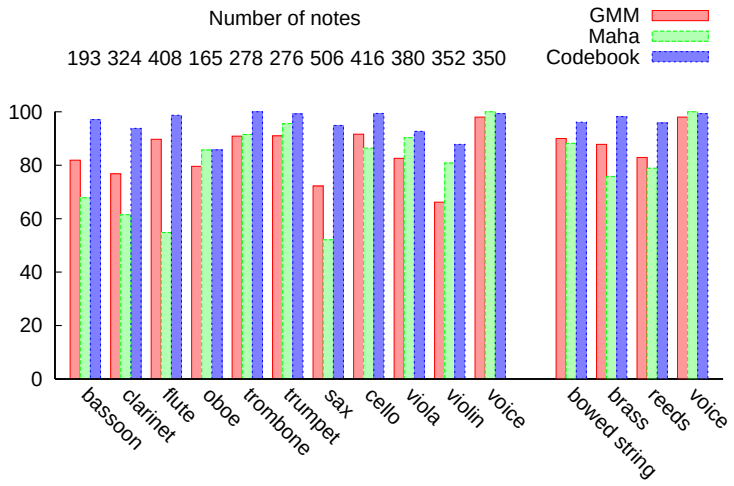


Figure 4.3: Average correct identification rates [individual and family] for the three classifiers in the configuration in which they perform the best (GMM: 40 clusters / 8 LSF, Maha: 30 clusters / 16 LSF, Codebook: 40 clusters / 24 LSF). Figures at the top correspond to the number of notes per instrument that have been considered for the experiments.

can be found in [CDS05a]. In Figure 4.1.6 are resumed the performances of the three systems in the configuration in which they perform the best. Using the same features, performances (individual and average correct identification rates) of the codebook to codebook distance is compared to a GMM classifier on one hand and to a K-means using a Mahanalobis distance on the other hand.

Both in terms of individual and average correct identification rates, the codebok to codebook distance gives better average and individual recognition rates than the other systems. Performances of the GMM / Mahanalobis based classifier are often comparable and always inferior.

It can also be noticed that whatever classifier is used, singing voice excerpts are correctly identified more than 95% of the time. For this reason, the design of a system using the LSF as features for the segmentation singing/voice music in recordings will be our next contribution to the SIMAC project.

4.1.7 Conclusion and Future work

The use of the pair LSF/k-means in a musical instrument identification context has been described. The definition of a distortion measure between two codebooks has been introduced and performances were evaluated for the classification between 11 classes of instruments. For an identical number of clusters, classification techniques based on the minimisation of the isotropic Euclidean distance performed better than classifiers taking the variances into consideration: at least, a gain of 11.6% in average correct identification rate was observed (Euclidean vs GMM) and at best, performances increased by 16.5% (Euclidean

vs Mahanalobis).

However, other experiments showed that LSF are highly sensitive to recording conditions, playing style or instrument brand: when models were trained using one database and tested using the other available database, performances dramatically decreased. Such algorithm behaviour sets the limit of the monofeature approach in musical instrument identification systems that pre-processing stages, taxonomic classification strategies and multifeatures systems attempt to tackle.

It is clear that a system similar to the one presented here provides limited applications for the SIMAC framework *as it is*. A generalisation or extension to instrument identification within polyphonic recordings has to be envisaged first. However, it has to be noted here that this problem is extremely difficult to tackle. The considerable amount of instruments encountered in modern music together with the lack of robust techniques for demixing polyphonic recordings set the limits of the training / testing approach. For these reasons, we are currently designing an algorithm that could be used for the identification of singing voice segments in music recordings: a general voice model (in essence, a codebook of LSF vectors) is used to decide whether voice is present or not in a given observation. The goal is to segment entire songs according to singing voice information, thus building a pre-processing stage for an artist classification module. Another research orientation (in collaboration with Andrew Nesbit, from QMUL) is concerned with the application of blind source separation techniques for the identification / characterisation of the lead instrument. Performances of the DUET [YR04a] and short-term Fourier transform based source separation algorithms are currently evaluated in the case of stereo mixtures.

Both these approaches reflect the predominant view within SIMAC, that is best to limit ourselves to a constrained and well-defined set of instrumental sounds to be identified in multi-instrumental recordings, instead of trying to tackle the extremely challenging issue of unconstrained instrument identification. In the following section we present this approach and show successful examples for the case of percussive sounds.

4.2 Percussion classification in polyphonic audio recordings using localized sound models

In this section we deal with the automatic transcription of percussive sounds mixed in polyphonic and multi-instrumental audio signals. That is, given a multi-timbral audio signal, the goal is twofold: to automatically *classify* its percussion sounds and to automatically determine their *positions* on the time axis.

For instance, snare drum sounds can show large variations in timbral characteristics. In automatic *isolated* sound classification [HPD03b], this is typically dealt with from a machine learning perspective: a sound model (roughly, thresholds for specific relevant signal features) is built from a large, diverse collection of labelled snare drum sounds. This model is subsequently used to assign labels to unknown instances.

However, in our framework, the temporal boundaries of the sounds to classify are unknown. A list of *potential* percussion sound occurrences must be first

extracted from the audio recording. Different rationales have been proposed to solve this issue. For instance, one may assume that percussion sounds are bound to occur in fixed-length regions around specific time-points, either sharp onsets [GTM93, GPD00a] or beats at the tatum level [GH01a, PK03b, DU04].

Dealing with polyphonic recordings raises an additional issue: percussion sounds are superposed with, and surrounded by, a high level of "noise", i.e. other instruments as e.g. voice, piano, guitars etc. Even worse, simultaneous occurrences of several classes of percussion instruments (e.g. kick + hi-hat or snare + hihat) may be encountered. To deal with this issue, existing literature proposes diverse research directions. Some advocate source separation techniques as Independent Subspace Analysis [DU04, FCL02b] or signal models as "Sinusoidal + Residual" (assuming that drums are in the residual component) [HK02, PK03b]. Noise reduction techniques, as RASTA [KVES01], are also suggested. Another option is to build sound models from a large collection of labelled "noisy" percussion instrument sounds extracted from polyphonic audio recordings [HSG04]. The main assumption in this method is that, in average on the training database, the noise shows considerably higher variability than the drum sounds.

The approach by [GPD00b, ZPDG02] also assumes that percussion sound characteristics show less variability than surrounding noise; however, this assumption is not made at the scope of a training database, but rather at the smaller scope of *individual audio recordings*. They design very simple, general sound templates for each percussive sound (actual *waveform* templates [GPD00b, ZPDG02],) and find the several sub-segments in the audio recording at hand that best match those templates (by means of a correlation function). This process is iterated several times and sound templates are gradually refined by time-domain averaging of the best-matching segment in the very audio recording at hand.

Our approach is to *combine* general, prior knowledge about the sound characteristics of percussion instrument families with on-the-fly acquired knowledge of recording-specific sounds. Instead of pursuing universally valid sound models and features [HPD03b, HSG04], unique, localized sound models are built for every recording using features that are *locally* noise-independent and give good class separation. Instead of actually synthesizing new waveform templates from the audio signal [GPD00b, ZPDG02], we tailor (in a gradual fashion) feature spaces to the percussion sounds of each recording.

Therefore, an onset detector yields N potential drum sound occurrences that are subsequently processed as follows:

1. Classification using general drum sound models
2. Ranking and selection of the $M < N$ most reliably classified instances
3. Feature selection and design of localized models using those M instances
4. Classification of the N segments using the localized models

As it turns out, our attempts at the automatic ranking and selection has not yet provided satisfactory results. Therefore, we corrected manually the output of steps 1 and 2 and provided only correct percussion sound instances to the feature selection step. Consequently, in this paper, we present a more focused evaluation of the localized sound model design, as well as a proper comparison

between general sound model and localized sound model performances. Using the corrected instance subsets, we investigate how the performance of the localized models evolve as increasingly smaller proportions of the data is used for feature selection and training.

Automatic techniques for instance ranking are currently being evaluated. Together with the evaluation of the fully automatic system they are the object of forthcoming work.

4.2.1 Data and features

The training data set for general model building consists of 1136 instances (100 ms long): 1061 onset regions taken from 25 CD-quality polyphonic audio recordings and 75 isolated drum samples. These were then manually annotated, assigning category labels for *kick*, *snare*, *cymbal*, *kick+cymbal*, *snare+cymbal* and *not-percussion*. Other percussion instruments like toms and Latin percussions were left out. Cymbals denote hi-hats, rides and crashes.

Annotated test data consists of seventeen 20-second excerpts taken from 17 different CD-quality audio recordings (independent from the training data set). The total number of manually annotated onsets in all the excerpts is 1419, average of 83 per excerpt.

Training and test data are characterized by *115* spectral features (averages and variances of frame values) and temporal features (computed on the whole regions), see [HSG04] and [Pee04] for details.

The experiments described in the remainder of this paper were conducted with Weka [1].

4.2.2 Classification with general models

In order to design general drum sound models, we first propose to reduce the dimensionality of the feature space by applying a *Correlation-based Feature Selection* (CFS) algorithm (Section 4.2.4) on the training data. From the total of 115, an average of 24.67 features are selected for each model.

This data is then used to induce a collection of C4.5 decision trees using the AdaBoost meta-learning scheme. Bagging or boosting approaches has turned out to yield better results when compared to other more traditional machine learning algorithms [HSG04].

4.2.3 Instance ranking and selection

The instances classified by the general models must be parsed in order to derive the most likely correctly classified subset. Several rationales are possible. For instance, we can use instance probability estimates assigned by some machine learning algorithms as indicators of correct classification likelihood. Another option is to use clustering techniques. Instances of the same percussion instrument, which we are looking for, would form the most populated and compact clusters while other drum sounds and non-percussive instances would be outliers.

However, as mentioned above, we went for a "safe" option: manually parsing the output of the general classification schemes. Using the corrected output, we

[1] *http://www.cs.waikato.ac.nz/ml/weka/*

investigated how the performance of the localized models evolved as increasingly smaller proportions of the instances selected from a recording were used to classify the remaining sound instances of the recording. Since dependence between training and test data sets is known to yield overly optimistic results, these test were performed by doing randomized, mutually exclusive splits on the complete collection of instances for each recording.

4.2.4 Feature selection

A collection of correctly classified instances from a recording are then used to build new, localized sound models.

Relevant features for the localized sound models are selected using a *Correlation-based Feature Selection* (CFS) algorithm that evaluates attribute subsets on the basis of both the predictive abilities of each feature and feature inter-correlations [Hal00]. This method yields a set of features with recording-specific good class separability and noise independence. The localized models may differ from the general models in two respects: they may be based on 1) a different feature subset (feature space) and 2) different threshold values (decision boundaries) for specific features. As a general comment, the features showed significantly better class-homogeneity for localized models than for the general models.

4.2.5 Classification with localized models

Finally, the remaining instances must be classified with the recording-specific (localized) models.

For this final step, we propose to use instance-based classifiers, such as 1-NN (k-Nearest Neighbors, with $k = 1$). Instance-based classifiers are usually quite reliable and give good classification accuracies. However, usual criticisms are that they are not robust to noisy data, they are memory consuming and they lack generalization capabilities. Nevertheless, in our framework, these are not issues we should be worried about: by the very nature of our method, instances are reliable, there are few of them and we explicitly seek localized (i.e. not general) models.

4.2.6 Experiments, results and discussion

Table 4.2 shows a comparison of the performance of the general and localized models applied to the polyphonic audio excerpts. The number of selected features is constant for the general models, but individual to each recording for the localized models. The classification accuracies of the localized models are determined using 10-fold cross-validation.

The number of features selected for the localized models is significantly less than for the general models. At the same time, the performance of the former is clearly superior. Maybe not surprisingly, this is resulting from the lesser variability of percussion sounds within a specific recording, which gives clearer decision boundaries in the feature space between instances of the different categories.

Doing feature selection on all sound instances of a recording (100%) should give what we consider the "ideal" feature subset, which should give optimal

Model	General		Localized	
	# feat.	Accuracy	# feat.	Accuracy
Kick	19	80.27	5.73	95.06
Snare	33	70.9	10.41	93.1
Cymbal	22	66.31	10.94	89.17

Table 4.2: Average number of features used and accuracy (%) for kick, snare and cymbal sound classification in polyphonic audio recordings, using both general and localized models.

Percentage	Kick		Snare		Cymbal	
	# features	Accuracy	# features	Accuracy	# features	Accuracy
50%	5	89.9	11.86	90.84	6.67	86.73
40%	5.1	88.42	8.71	87.88	7.13	85.35
30%	3	86.6	5.71	82.96	3.57	84.72
20%	2.5	85.51	4	77.22	3.4	79.4
10%	1	77.92	1.71	73.34	1.27	71.53

Table 4.3: Average number of features used and accuracy (%) for kick, snare and cymbal sound classification using decreasing proportions of correct instances to select relevant features and perform 1-NN classification.

performance (noise-independence and class separation) on the individual recordings. Figure 4.4 shows the average classification accuracy of the kick, snare and cymbal models, using the optimal feature subsets for each localized model. The training-test data splits are repeated 30 times for each reported training data set percentage.

We see from the figure that the accuracy never drops down below that of the general sound models (marked by dotted lines). It seems like the performance makes a significant drop around 20% – 30%, indicating a sort of threshold on the minimum number of instances needed to permit successful classification. This proportion corresponds to about 17 – 25 samples. Further studies have to be done to establish whether it is the relative percentage or the approximate number of samples that is significant for the performance of the localized models.

In practice it is not possible to know the optimal feature subsets, as feature selection must be performed on a reduced data set. Table 4.3 shows average classification accuracies together with the average number of selected features for kick, snare and cymbal models, using truly localized features.

There is a slight loss of performance from localized models with optimal feature subsets (Figure 4.4). Using 30% of the instances, the accuracy decreases 7.3% for kicks, 7.57% for snares and 1.17% for cymbals. We observe that a reduction in the amount of training instances greatly effects the feature selection. Besides a general decrease in number of selected features, the variation in types of features selected for each recording can be high.

What is not evident from the tables, is the variability of the performance among individual recordings. At one extreme 96.72% accuracy is obtained using only 1 feature and 10% of the complete data set. When comparing to classifica-

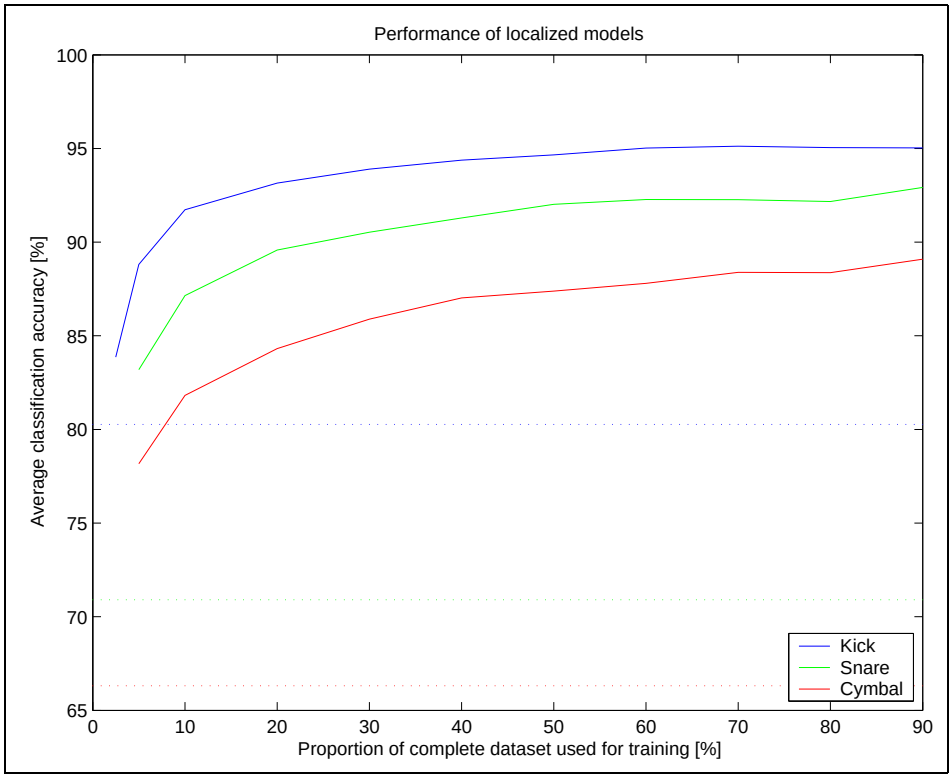


Figure 4.4: Accuracy for kick, snare and cymbal sound classification using the optimal feature subsets and decreasing proportions of correct instances to create the localized models. The dotted lines mark the accuracies obtained with general models.

tion with general models, it appears that recordings having the least successful localized models are also least favorable for classification with general models.

Also, it is important to notice that relevant features for localized models usually differ from one recording to the other, which justifies the proposed method. Let us focus, for instance, on single-feature based kick models. Depending on the specific recording at hand, some used e.g. the mean of the frame energy values in the 1st Bark band, others the mean of the 3rd spectral moment in successive frames, or other features. Snare models used e.g. the mean of the frame energy values in the 11th Bark band or the mean of the 4th MFCC in successive frames, etc. Cymbals models used e.g. the mean of 9th MFCC in successive frames or the mean of frame spectral flatness values, etc.

4.2.7 Conclusion and future work

In this section we propose the design of feature-based percussion instrument sound models specialized on individual polyphonic audio recordings. Initial classification with general sound models and parsing of their output provides a reduced set of correctly classified sound instances from a single recording. By applying a a feature selection algorithm to the reduced instance set we obtain the reduced feature sets required to design recording-specific, localized sound models. The localized models achieved an average classification accuracy (and feature dimensionality) of 95.06% (5.73) for kicks, 93.1% (10.41) for snares and 89.17% (10.94) for cymbals, which represents improvements of respectively 14.79%, 22.2% and 22.86% over general model classification accuracies. We also showed that the choice of relevant features for percussion model design should be dependent, at some extent, on individual audio recordings.

Part of future work is to implement a semi-automatic percussion transcription tool based on the approach presented here. Our results are encouraging, but we need to process more and longer recordings to claim that the method is general and scales up well. More effort has to be put into determination of reliable estimators of general model classifications. We must also consider the influence of noisy data in localized model design. Another direction for future work is to explore whether ISA, RASTA or 'Sinusoidal + Residual' pre-processing can improve the classification performance.

Alternatively, instead of attempting to identify percussive sounds in the mixture, we can try to describe them as a function of abstract metrics that somehow characterize the relevant attributes of this type of instrumental sounds. It is in this direction that the work in the next section focuses on.

4.3 Percussion-related semantic descriptors of music audio files

The approach to music content extraction presented in the previous sections is one that we can qualify as *transcriptionist*. According to this approach, describing music content equates to extracting a score-like representation of the original audio. In contrast with this transcriptionist view, we suggest here that a *descriptionist* approach can be used, as advocated elsewhere by Martin et al. [MEV98], or Carreras and Leman [CL00]. The descriptionist approach is an ecological and user-centred way of addressing the description of music contents

from audio files. It is an ecological approach because the research context is that of a system in use, the structure and functionalities of which will pose specific problems and will shape the knowledge structures of the users. It is a user-centred approach because the attempted solutions spring from the user needs and requirements, and not by the need to satisfy a pre-existing musical theoretical construct.

From the work on the previous section we know that focusing on percussive sounds helps us gain some insights into the instrumentation-related contents of polyphonic and multi-instrumental recordings. According to this approach, we advance several percussion-related descriptors that do not correspond to solid musical theoretical entities, but to entities that are (or can be) represented in the minds of the users of music information retrieval systems. Because of that, they can be exploited, taken one by one or combining them synergistically, to define and refine query and retrieval operations of music files. These descriptors we have named *percussion index*, *percussion profile*, *kick-snare crossings*, and *percussivity*.

4.3.1 Background

Although percussion has been traditionally neglected in music analysis research, in the last two years we have witnessed a growing wealth of papers focusing on it, mainly with a focus on transcription. Goto and Murakoa [GM94] studied drum sound classification in the context of source separation and beat tracking [GM95b]. They implemented an "energy profile"-based snare-kick discriminator, though no effectiveness evaluation was provided. More recently, Zils et al. [ZPDF02] reported very good performance rate at identifying kicks and snares in songs by means of a technique of analysis and incremental refinement of synthesis that was originally developed by Gouyon [GPD00c]. Jørgensen [Jør] attempted to use cross-correlation between sound templates extracted from isolated sound recordings and realistic drum-kit recordings. Using this technique only kicks and snares seem to be detected with some reliability. A very different motivation has been that of Kragtwijk et al. [Kra01] who have presented a 3D virtual drummer that re-creates with synthetic images the playing movements of a real drummer after analyzing the audio input coming from a real performance. Unfortunately, the audio analysis part of the system was finally underdeveloped.

Riskedal [Ris02] developed a system that combined a drumloop-adapted onset detection procedure (taken from Klapuri [Kla99]) with an adaptation of Independent Component Analysis [and95], a source separation technique. Though the results seemed promising, only a few examples were presented by the author.

Orife [Ori01] developed, also without providing systematic evaluation, a tool for extracting rhythmic information from loops and songs using Independent Subspace Analysis (ISA), a technique for source separation discussed by Casey and Westner [CW00]. More recently, Fitzgerald et al. [FCL02a] have also taken advantage of ISA and of some additional sub-band pre-processing, and reported a success rate of 89.5% when transcribing a database of 15 drum loops containing snare, kick and hi-hats. When knowledge about the sources was incorporated, then a Prior Subspace Analysis technique [FCL03] allowed them to achieve 92.5% of correct identifications. When they extended the technique to music (i.e. mixtures of pitched instruments and drums) [FLC03] they got 89.3% of correct identifications for a database of 25 drum loops. A possible

drawback, though, is that the system needs human intervention, and is fit to a specific kind of sounds. Uhle et al. [UDS03], also using ISA and descriptors like percussiveness, noise-likeness, spectral dissonance, spectral flatness, and the third order cumulant for classifying the segments after ISA decomposition, reported 95% percent of correctness for a database of only 9 music titles. Paulus and Klapuri [PK03a] have used acoustic models and N-gram based models that allowed getting acceptable performance rates, though the approach seemed to be not general enough. More recently [PK03c], they have improved the system by including rhythm information as an additional cue for label assignment. The system has been tested with 359 MIDI songs (that were rendered into audio), yielding up to 86% of correct decisions. Virtanen [Vir03], using sparse coding, i.e. another source separation technique, achieved 66% of effectiveness when separating the kick and the snare from the rest of musical mixes from a database of 100 MIDI polyphonic songs (that were rendered into audio). A different approach, based on Parametric Vector Quantization, has been presented by Wang et al. [WTAV03]. Their system features computed on 3 sub-bands for clustering those events that could reliably be considered as "percussive". Small scale evaluations yielded an average of 84.5% of correctness. Steelant et al. [vSTD+04] have also addressed the transcription problem and their first results classifying kick and snare sound slices extracted from music CDs and MIDI songs achieved recall rates ranging from 85% to 96%, depending on the type of Support Vector Machine they used for inducing the classification models.

One major criticism in most of the existing research is that of using small databases (though in the current status our research also suffers from the same drawback). Another one is that of disregarding performance evaluations that are based on independent samples (i.e. not used during the analysis or training phase). As we discussed elsewhere [HDG03a], the estimations can therefore be as much as 10% overoptimistic when training and testing is done using the same sample of observations.

Our approach, in contrast to the previous ones, has purposely skipped sound separation techniques in order to keep computation requirements as low as possible, and has not focused on absolute transcription of percussive events, but on the creation of semantic descriptors related to them. More details can be found in [HSGP05].

4.3.2 Pre-processing

Most of the descriptors presented here are computed after an onset detection and percussion segment extraction process. We use an algorithm adapted from Klapuri [Kla99]. After detecting onsets, audio slices of 100 milliseconds starting from the onset position are segmented. A high-pass filter is then applied to the extracted slices though the original version is also kept for being used by some of the classifiers used for the assignment of instrument labels.

4.3.3 Percussion Index

The *Percussion Index* (PI from now on) estimates the amount of percussive events that can be found in a music audio file, allowing a user to query for music titles containing from "no percussion at all" to "a lot of percussion".

In order to classify the audio slices as containing a percussion sound or not, we take advantage of a long list of low-level timbre-related descriptors including MFCCs, energy in Bark bands, spectral centroid, skewness, flatness, their variances, etc. We use them as input for with inductive modelling techniques that take the decision on the presence/absence of percussion in the extracted segments. We have found several meta-learning approaches such as bagging [Bre96] or boosting [Sch99] yielding the best results (compared to other more traditional options such as neural networks, support vector machines, simple decision trees, or lazy learning). Once a dichotomic decision has been taken for each extracted segment, the PI is computed as the ratio between the events containing percussion against the total number of events.

A ground-truth evaluation database has been set up in order to estimate the reliability of the index for the task of song description. The database contained twenty-nine 20" segments extracted from songs coming from all kind of genres and epochs (from nineteen-forties to the current year). This amounts up to almost 1500 events to be classified. The objective evaluation of the PI was done by comparing the PI obtained using the twenty-nine songs that were manually annotated with the PI that was obtained when the songs were automatically segmented and classified. The Pearson correlation coefficient was of 0.67 ($p < 0.001$), which is an evidence of the validity of the automatically computed PI (i.e. it is computing what is intended to be computed, under some error tolerance), but also is a hint that there is still some room for improvement.

The nature of the PI makes it quite robust to local misses of onsets. Even if the onset detector misses some of the percussive or non-percussive events, the computed value still keeps overall meaningfulness for the whole song. In addition, a certain amount of misclassifications can also be accepted. The robustness of the PI is even increased once we convert the real value into a discrete one: although the original computation of the PI yields a real-value which is bounded between 0 and 1, a quantization is needed in order to make it usable in the real-world. A JND or just noticeable difference can be asked for and, even though the evaluation is still being finished, it seems that it will yield between four and five discrete labels for a final user to exploit consistently the underlying concept (*no percussion*, *low amount of percussion*, *fair amount of percussion* and *lots of percussion*).

Bartok: Music for Strings, Percussion and Celesta IV Allegro molto	0.22
Mancini: Pink Panther Theme	0.46
Pet Shop Boys: It's a Sin	0.56
The Beatles: Help!	0.66
Portishead: Strangers	0.91

Table 1. An example of sorting popular titles according to the raw PI value (it was computed using only an excerpt of 20 seconds).

4.3.4 Percussion Profile

The *Percussion Profile* (PP from now on) is a further refinement of the PI. It has been devised in order to yield an index that gives relative information

about specific classes of percussion instruments. The PP makes possible to describe songs as "having lot of cymbals" or "being short on snares". The practical requirement for achieving a reliable PP computation is the existence of reliable models for the specific classes we are interested in (for example, membranes versus plates and finer distinctions). Hence, the problem addressed by this descriptor is much more complex than that of the PI, as here we need to discriminate between, for example, a snare and a hi-hat, in the context of a mixture containing also harmonic sounds. Even worse, sometimes we will find simultaneous occurrences of several classes of percussion instruments (i.e. kick + hihat, snare + hihat).

Three different strategies can be envisioned to achieve the computation of the PP: raw class modelling, standard pre-processing of the slices (i.e. using filtering or enhancements, or trying reverse-engineering of noise-reduction techniques), and source separation prior to attempting class decisions. We have opted for the raw classification, though a high-pass filtering is done by the classifier that is specialized in the cymbals. The effective computation of the PP is achieved by a hierarchical classifier that first separates percussion slices from non-percussion slices, then, percussion slices are divided into membranes and plates, then single hits or combinations of sounds are detected, and finally the kick and snare are further separated. All the computed decisions are then integrated and the final label assignment {kick, snare, cymbal, kick+cymbal, snare+cymbal, no-percussion} is computed. Each classifier uses a different set of features and the error rate of them, using a holdout database taken from a different song collection than the one used for learning the class models is kept under 15%. An absolute objective measurement (i.e. comparing the automatically assigned labels with manually annotated labels) is under way.

4.3.5 Kick-Snare Crossings

Kick and snare hits do not usually happen at the same time as they contribute to keep a contrastive running beat pattern. Their roles are different, and the ways their sounds are organized along time contribute to define specific rhythm and timbre patterns. For example, in the 80's pop music, the snare was to be found in every downbeat of a 4/4 bar (i.e. 2^{nd} and 4^{th}), whereas in reggae music the snare can be usually found mostly in the 3^{rd} beat. Hence, the changes from kick to snare are expected to be different, on average, for each of these two genres. The average number of changes from kick to snare and the other way round is computed as the Kick-Snare Crossings (KSC) descriptor. As the current formulation of this descriptor does not yield a true semantic descriptor we are searching for ways to map it into a proper concept to be exploited for end-users of a MIR system.

4.3.6 Percussivity

Contrasting with the previous ones, the it Percussivity descriptor includes a heavy perceptual quality in its definition. According to the previously presented descriptors, two songs can be quite similar, but one of them having the percussion parts much more prominent than the other (for example, by mixing percussion sounds to be "slapping in the face" when compared with the rest "backing" instrumentation). Another example can be found when comparing

a given rhythm pattern played with brushes or played with sticks. In the first case the percussive sensation is lower than in the second case. Percussivity can also be found whenever no percussion instruments are played at all. That is the case of a string section excerpt played *pizzicato*, which will sound more percussive than the same excerpt played *bowed*. We suggest quantifying this kind of sensation by means of defining a "percussivity" descriptor that is computed by adapting an approach taken from research on the perceived impulsiveness of environmental sounds [Ped01]. Given the nature of the percussivity descriptor, it can only be evaluated by means of a listening experiment using human subjects.

4.3.7 Discussion

The descriptors we have presented here are intended to complement other more usual descriptors of rhythm, tonality, melody, structure, or even those belonging to a more arcane *complexity* facet. All of them are being integrated in the prototypes developed under the SIMAC project. We believe that a truly usable MIR system for music audio titles needs to exploit descriptors addressed to most of the musical facets that can be computed, and this exploitation has to be done according to the knowledge level of the intended users. It is usually the case that their knowledge level does not have much to do with the constructs managed by music theories. Our descriptors seem to be valid and reliable, and keep a kind of coarseness that makes them learnable and usable in practical situations.

We still have important open issues such as the conversion of the KSC into a truly semantic descriptor, or the improvements of the PP. There is also the need to perform a truly large-scale evaluation using an independent (i.e. containing instances that were not used during the learning phase of the instrument models) test database.

A final additional refinement that is worth to be considered is that of distinguishing between *percussivity* and *percussiveness*. The later can be defined as a quality or possibility of a sound for being percussion-like (see Uhle et al. [UDS03] for a formal definition that is not far from our concept). For example, there are synthetic sounds that, even though they are not being created by percussion instruments, can be considered as playing the role of them and/or mocking them (i.e. a blast of filtered noise, or some noises made by striking the keys of wind instruments). This kind of percussiveness would probably deserve a new descriptor too.

4.3.8 Conclusions

Extracting semantic descriptors from music audio files does not necessarily call for a transcription. Scores are only one of the multifarious semantic devices we can take advantage of when trying to describe music contents and most of the semantic descriptors that can be managed after transcribing a musical piece can only be exploited by people having been educated under western-music traditions. In order to find truly usable semantic descriptors for popular music retrieval systems (i.e. Kazaa, iTunes, Napster) we have to quest for extracting knowledge from users.

For the particular case of instrumentation-related description, the proposed descriptors can be considered the first captures we have got in this quest. There

are many particularities of percussive sounds in a mixture that allow us to characterise them even from within the "noise" introduced by other instruments in the ensemble. However, this approach is not so straightforward for other types of sounds, whose harmonic content tends to overlap with that belonging to other instruments in the mixture. Under the SIMAC project we have started exploring possible solutions to this problem under the research accolade of blind signal source separation. In the next section we present the current status of our work in this area and our new relevant research trends.

4.4 Source separation of stereo music signals

Generally, blind source separation (BSS) is a framework for separating sources from one or more observed mixtures when little or no information of the sources is known in advance.

An instantaneous mixing model is generally assumed, so that sources can have different attenuations in each mixture but no reverberation is present. There is some provision in this work for anechoic mixtures so that sources can also have delays in the mixtures, but this is not emphasised.

There are several interrelated aims to attempting to perform blind separation of sources from stereo music signals:

- to extract musical sources in an appropriate way so that the source can be identified by an instrument identification system (for example [CDS05b])
- to identify and extract the predominant source from a mixture
- to investigate source separation algorithms for extraction, enhancement and suppression of sources and how they can help solve the previous aims

In this section, we outline the research that has so far been undertaken to satisfy these aims.

4.4.1 Overcomplete blind source separation

If there are more sources present than mixtures observed, then the problem of determining the mixing parameters and sources is overcomplete. This means that the sources and mixing parameters cannot be uniquely determined using traditional matrix inversion methods. The general overcomplete model for BSS is this:

$$\begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} s_1 \\ \vdots \\ s_n \end{bmatrix}$$

where each of the source signals s_i are statistically independent and $n > m$ (the number of sources is greater than the number of observations).

Much work on solving the overcomplete case assumes that the sources have a sparse representation. Each source can be represented as a linear combination of elements from a large set, where most of the coefficients in the linear combination are zero or close to zero:

$$s = \sum_{i=1}^{N} a_i \phi_i$$

where the a_i are mixing coefficients, most of which are zero or close to zero, and ϕ_i are called *atoms* or *elements* from a (possibly overcomplete) dictionary of size N.

It has been shown that this property of sources can lead to better estimation of mixing parameters and a better quality of source separation [Car98, ZP01]. Such attempts typically project the sparse representation onto some other space and then form clusters whose parameters correspond to the mixing parameters.

We implemented several known algorithms for overcomplete blind source separation of audio signals and assessed their suitability for stereo music signals. All methods discussed make the assumption that sources are disjoint in the time-frequency plane, that is, each time-frequency point carries energy from at most one source.

4.4.2 The degenerate unmixing estimation technique (DUET)

The first method is the degenerate unmixing estimation technique (DUET) [YR04b]. It assumes an anechoic mixing model. Each source present has unique attenuation and delay parameters. For each time-frequency point, an estimation of its attenuation and delay parameters is computed. A histogram of all estimates is produced: The number of peaks indicated the number of sources, and the coordinates of each peak gives the corresponding source's estimated amplitude and delay parameters. From this, a time-frequency mask can be generated for each source, and a source is demixed by applying the masks to one of the mixtures.

4.4.3 Azimuth discrimination and resynthesis (ADRess)

The second method is the azimuth discrimination and resynthesis (ADRess) technique [BLC04], similar to DUET. It assumes an instantaneous mixture (no delays). The signal is transformed by short-time Fourier transform, and for each frame, a scaled version of one mixture is subtracted from the other. For a particular scaling factor, this cancels sinusoids corresponding to the source, and the scaling factor indicates the location of the source in the stereo plane. Given this, the source's magnitude spectrogram can be determined and the source recovered.

4.4.4 Sparse mixtures of Gaussians

The third algorithm models each source as a mixture of Gaussians (MoG) to approximate a non-Gaussian distribution [DM04]. It assumes the sources are sparse and operated in the modified discrete cosine transform (MDCT) domain. A two-state model is assumed, in that at any time, each source is inactive or at most one source is active. In the general formulation it is computationally intractable, but simplyfying assumptions can be made because most of the Gaussians in the mixture have negligible weights. Thus the mixture is truncated so that the number of Gaussians is linear in the number of sources and an expectation maximisation algorithm determines the parameters of the audio mixture. The sources are recovered using minimum mean squared error, followed by inverse MDCT.

4.4.5 Experiments

Experiments have been performed to assess the suitability of each of these approaches in separating sources from stereo mixtures of synthetically mixed music and studio recordings from CD. Testing was performed on the following stereo signals:

1. synthetic mixture of two speech sources from the TIMIT speech recognition database
2. synthetic mixture of three musical instruments (piano, bass, drums)
3. synthetic mixture of three musical instruments with delay
4. CD recording of four musical instruments (John Coltrane)
5. studio recording (*Personalized Perfection* by Another Dreamer, from the Creative Commons database)

Quantitative results were computed by applying each of the DUET and MoG algorithms to the mixtures.[2] (The ADRess algorithm was implemented at a later stage and so was applied to only a few of the test cases.) However, at this stage, it is more meaningful to summarise the results qualitatively.

The results on music, so far, are not practically useful due to much interference and overlapping extracted sources. However, the methods can be useful if sources are very sparse, that is, for speech or "unrealistic" music (in which the sources did not have a high degree of harmonic overlap, and which were mixed synthetically). The major issue here is the assumption of time-frequency disjoint sources: Although this constraint allows for neat and appealingly simple algorithms, it is unrealistic for music signals. The reason is that musical sources, by design, usually have a high degree of harmonic overlap (moreso than speech).

Interesting results were obtained in some tests with the MoG algorithm, in relation to extracting the predominant source. For when it was specified to the algorithm that two sources were present, in some cases the extracted "sources" were, respectively, the predominant source and "everything else". Identifying the predominant source was based on subjective preference. In particular, singing vocals or an instrument playing a melodic solo were classified as predominant (see Section 4.4.6).

A downfall of the DUET and ADRess techniques is that they require much manual parameter tweaking. A set of parameters which works well for one musical excerpt may perform badly on a different excerpt. Furthermore, estimation of the mixing coefficients is currently performed by manually selecting peaks on a histogram. A weighted k-means algorithm has been used to try to deduce the peaks automatically, but this has, so far, been unsuccessful.

[2] In particular, the following performance measures were used:

preserved signal ratio (PSR) measures how much of the true source is retained in the estimated source

signal to interference ratio (SIR) measures the amount of unwanted sources which have been extracted along with the intended source

signal to distortion ratio (SDR) gives an indication of the total relative distortion

sources to artifacts ratio (SAR) indicates the relative distortion introduced purely by the algorithmic separation process

4.4.6 Further work

We are currently investigating these techniques as a preprocessing stage for a system which identifies a singing voice. In this case, enhancement or separation of the voice must preserve its timbre well enough so that input to an identification system is valid. An extension of this idea is that of extracting or enhancing the predominant source, and this raises the questions: What is a predominant source? Is there some correspondence to variance within the Mixture of Gaussians framework? Is there a correspondence to a source's location in the stereo plane? Indeed, predominance of a single source is highly subjective, and some sort of definitions need to be made to clarify this nebulous concept.

Each of the methods assumes that all sources have the same variance. Therefore, pre-filtering (for example, by the use of equivalent rectangular bandwidth filterbanks [MG96]) may be able to mitigate against assumption.

Another method for improvement of time-frequency techniques is adaptive basis selection to give a better representation of the signal as it evolves over time. Also, investigations into sparse signal representations, for example, how to adapt DUET so that MDCT might be used for sparse representation, may be useful.

The time-frequency disjoint assumption of the sources (and the corresponding binary masking) is an unrealistic hypothesis on the sources. Therefore, progressive masks will be investigated so that more than one source can occupy each point in the time-frequency plane.

Finally, structured source priors based on independent subspace analysis and hidden Markov models [VR04] can give structured models to musical sources so that to reduce the amount of "blindness" that typical BSS algorithms assume.

4.5 Conclusions

In this chapter we have presented research towards the generation of a semantic description scheme for music instrumentation and timbre. This differs from previous work on "textural" description, on that we attempt to truly characterise the content related to instrumentation and not only the acoustic properties of the polyphonic sound in the recording (as work on fingerprinting tried to do before us).

Towards this end we have presented an approach to the identification of instruments in mono-timbral recordings, using line spectral frequencies and K-means classifiers. While showing relative success we have seen that an alternative view needs to be taken to extend such a methodology to multi-instrumental recordings.

One possibility is to limit our instrument identification task to a subset of instruments whose attributes in the signal are identifiable even in the presence of "noise" introduced by other instruments. As an example we have shown how percussive sounds can be successfully recognised in complex mixtures, using an approach based on sound localization.

Another possibility, is to renounce altogether to the task of instrument identification and concentrate instead on describing the timbrical contents of the signals as a function of a particular set of characteristics of known sounds. To illustrate this, we have again decided to describe percussive instruments in multi-

instrumental signals, but this time as a function of abstract concepts that can be associated to attributes that users look for in music retrieval systems. This novel approach is at the core of the SIMAC system for all musical dimensions including timbre.

Finally,we have shown the status of work in progress towards the selective separation of sources in the mixture, as an attempt to "pre-process" the recorded data in a way that is useful to instrumental analysis, even having obvious implications for unresolved tasks such as melody estimation.

Chapter 5

Describing Intensity

As seen in previous chapters, automatic extractors of music metadata such as tonality and rhythm are valuable building blocks for advanced music retrieval applications that may provide functionality such as automatic playlist generation and music recommendation. Today, most commercial systems are based on editorial metadata and content descriptions that are manually annotated, either by experts or by a community of users. Content-based retrieval and automatic metadata extraction is a natural next step.

This work, based on [SGH04] presents a descriptive study of perceived intensity in popular music, with the goal of establishing an objective model of subjective intensity built from low-level descriptors extracted from the audio data.

Intensity in music, or *the sensation of energy we get from listening to music*, is a concept commonly used to describe music content. Although intensity has a clear subjective facet, we hypothesize that there exists a basic definition that can be objectively evaluated, i.e. a certain set of salient features determining intensity that most people agree on. We will establish an objective intensity model by doing the following:

1. Design an intensity taxonomy, where each category is accompanied by textual descriptions that communicate the desired interpretation of subjective intensity to the listener. This provides us with a vocabulary for talking about intensity in music.
2. Show that people perceive intensity in a fairly consistent manner, given the intensity taxonomy. We establish this by means of a listening test, where subjects are asked to assign intensity category labels to excerpts of music recordings.
3. Measure salient features of the audio data using appropriate signal processing that yields low-level descriptors. We will use the terms feature and descriptor interchangeably in this chapter.
4. Model intensity by applying a pattern recognition algorithm to the excerpts, represented by feature vectors and the assigned intensity category labels. Unknown music is classified using the resulting intensity model.

A similar study was performed by Zils and Pachet [ZP03], where perceived intensity was divided into four categories named from "low" to "very high". They also performed a listening test, and relied mainly on their Extractor Discovery System (EDS) software to discover new low-level descriptors correlated with intensity. Their model achieved reasonably good accuracy, but provided little insight into how the salient features of intensity interact. Our study reinforces many of their findings, as the data from our listening test is contributed by a different and larger population.

5.1 Semantic description of subjective intensity in music

5.1.1 Designing an intensity taxonomy

Perceived intensity in music is not a well-defined concept. It is not purely determined by a long-term loudness sensation, but may also be connected to

tempo, instrumentation, rhythmic complexity, etc. We want, however, to model the sensation of intensity as it is, independent of other concepts. Great care was therefore put into avoiding undesirable connotations in the textual descriptions of the intensity taxonomy, in order not to bias the subjects of our listening test. Words and sentences relating to concepts such as volume, tempo and emotion were specifically avoided.

The taxonomy was defined as follows:

Wild Marked by extreme lack of restraint or control; intensely vivid. Synonyms: *intense, manic, fiery.*

Energetic Possessing or exerting or displaying energy. Synonyms: *lively, sparkling, raucous/rowdy, exciting.*

Moderate Being within reasonable or average limits; not excessive or extreme. Synonyms: *laid-back/mellow.*

Soft Having or showing a kindly or tender nature. Synonyms: *gentle, soothing, calm/peaceful.*

Ethereal Characterized by lightness and insubstantiality; as impalpable or intangible as air. Synonyms: *detached, hypnotic, unreal.*

The definitions are taken from WordNet[1], an online lexical reference system. An exception is the definition of *soft*, which has an undesirable connotation related to volume. We used instead the definition of *gentle*, one of the listed synonyms.

The synonyms, and two of the category names, can be found in the mood taxonomy of *All Music Guides*[2], which also provides lists of reference albums and music titles for each mood. This connection is interesting, as it provides us with a second set of objective data for future evaluations of the final intensity model.

5.1.2 Gathering objective intensity data

An objective data set of music recording excerpts, sorted into intensity categories, was gathered through a listening test. We implemented a forced-choice style, Internet-based survey requiring only a JavaScript-enabled browser and MP3 playback software of the participants. A screenshot of the survey interface is shown in Figure 5.1.

In the survey, subjects were presented with a sequence of short excerpts of music and instructed to assign excerpt to one of the five intensity categories. The excerpts were presented in random order, reducing the risk of carry-over and order effects, i.e. where the intensity of a music excerpt may effect the perception of intensity in the next. The subjects were encouraged to spend between 15 and 20 minutes doing the survey. To ensure a stable listening environment, the subjects were led through a sound level calibration procedure and instructed to use headphones.

For audio data we selected 150 popular music recordings, spanning a wide range of genres and (as far as we could tell) intensity levels. A 20 seconds long excerpt was selected, 30 seconds from the beginning, in order to get more

[1] http://wordnet.princeton.edu/
[2] http://www.allmusic.com/

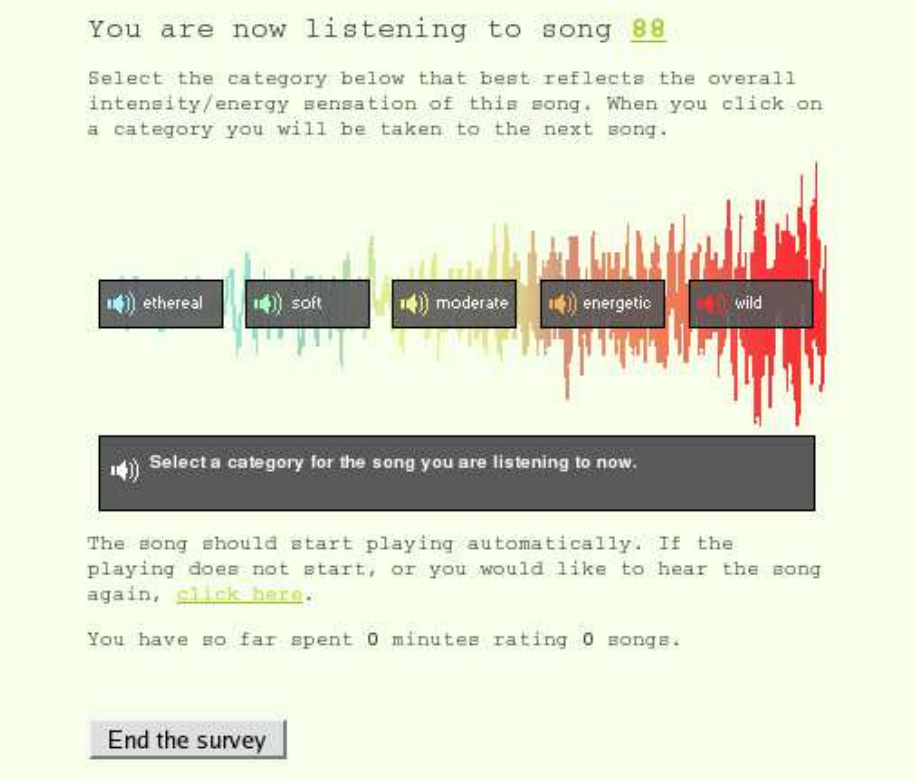


Figure 5.1: Screenshot of RateIt!, our Internet-based listening test.

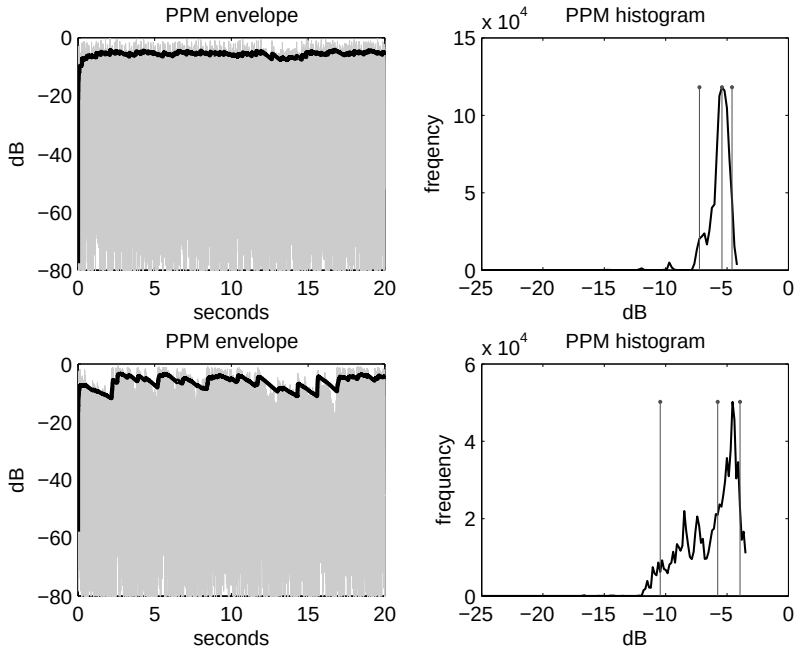


Figure 5.2: PPM envelopes and histograms for high intensity music (top) and low intensity music (bottom). The median, 5^{th} and 95^{th} percentile are marked onto the histograms.

homogeneous segments. (Avoiding lengthy intros etc.) Each excerpt was additionally audited to reveal undesirable characteristics, such as sudden changes in instrumentation or tempo halfway, and possibly replaced.

The subjects were mainly music students, and music technology students and researchers, from four universities in Spain and Norway, and must therefore be considered as trained listeners.

5.1.3 Extracting low-level audio features

The following feature extraction functions were implemented in MATLAB®.

Dynamics related descriptors

The dynamics of the sound was estimated for two time resolutions, coarse-grained using Root Mean Square (RMS) calculation and fine-grained using Peak Program Meter (PPM) calculation. Both produce a time-domain envelope in dB. We used histograms to approximate the distribution of dB values in the segment, and described the shape of the distribution in terms of percentiles, centroid, spread etc. Figure 5.2 shows PPM envelopes and distributions for two signals of different intensity, with the extracted features marked.

The RMS values were computed for 50 ms long, non-overlapping frames. Our PPM implementation is similar to the one described by Zölzer [**?**]. For stereo signals, the RMS and PPM envelopes were averaged prior to feature

extraction. The extracted RMS and PPM histogram descriptors were median, 5^{th} percentile, 95^{th} percentile and 90^{th} inter-percentile range, as well as centroid, spread, skewness and kurtosis (excess).

Spectral descriptors

Spectral descriptors were computed using the Short-Time Fourier Transform (STFT) with a frame length of 23 ms, 50% frame overlap and a Hanning windowing function. Descriptor values for the entire segment were modelled as the mean and variance of the individual frame values. The spectrum magnitudes were averaged prior to feature extraction in the case of stereo signals.

The extracted spectral descriptors were spectral centroid, spread, skewness and kurtosis (excess), as well as spectral flatness.

Long-term loudness estimates

Two estimates of long-term loudness were computed from the audio data.

In studies by Soulodre [Sou], and Skovenborg and Nielsen [SN04], the *equivalent sound level* (L_{eq}) measure with the *Revised Low-frequency B-weighting* (RLB) has shown to be a reliable, objective loudness estimate of music and speech. Defined as

$$L_{eq}(\text{RLB}) = 10 \log_{10} \left(\sum_{n=1}^{N} x_W[n]^2 \right) \tag{5.1}$$

where $x_W[n]$ is the frequency-weighted digital waveform and N is the total length of the signal, our implementation is similar to the one described in [Sou].

Skovenborg and Nielsen [SN04] introduced a new loudness model, LARM, optimized for computation of long-term loudness estimates of non-stationary signals. LARM is based on the asymmetrical low-pass filtering of the PPM, combined with RLB frequency weighting and power mean calculation. It achieved best overall performance in their evaluations when compared to traditional loudness models.

5.1.4 Modelling intensity in music

With a total of 28 low-level features and up towards 150 instances in the training data set, we risk over-fitting the intensity model. It is therefore necessary to reduce the number of features prior to intensity class modelling, using a suitable feature selection algorithm. *Correlation-based Feature Selection* (CFS) has previously shown to excel for feature selection in the context of percussion instrument sound modelling from low-level audio features.

We have made no assumptions about salient features prior to the analysis. Therefore we would like to have a transparent model of the intensity categories, where the significant features can readily be identified and inspected. *C4.5 decision trees* are suitable for this purpose, as the emergent tree structure provides a clear representation of decision boundaries in the feature space.

Both feature selection and the class modelling described in this section was performed with the WEKA[3] data mining software, in wide-spread use throughout the MIR research community.

[3] http://www.cs.waikato.ac.nz/ml/weka/

Feature subset	Accuracy
RMS	28.4%
PPM	47.8%
Spectral	67.2%
Leq(RLB)	40.3%
LARM	43.3%
Combined CFS	62.7%

Table 5.1: Performance of various feature subsets.

5.2 Results and discussion

The survey was operational for two weeks and resulted in over 3500 intensity category ratings, an average of approx. 23 ratings per music excerpt. The collected data was filtered according to several constraints in order to remove unreliable and possible erroneous contributions, thereby increasing its quality. Since the subjects go through an adjustment phase as they become familiar with the task, their initial contributions are likely to be "out of tune" with the ones following. The first five ratings of each subject were therefore discarded. Ratings that took less than five seconds or more than 100 seconds were also discarded, as this may indicate a lack of focus and possible fatigue.

Most of the 150 music excerpts display a unimodal distribution of ratings among the five intensity categories. Near half of them converged nicely toward one single category. The intensity model was built on a subset of the most consistently rated excerpts, as these are most likely to represent an objective consensus on perceived intensity in music. Excerpts were accepted into the training data set if they had received more than 10 ratings, and if one category had received between 70% and 100% of the ratings. 67 excerpts were selected this way. The rest of the music excerpts formed the test data set that we used to evaluate the final intensity model.

5.2.1 Evaluation of feature subsets

Table 5.1 shows the classification accuracy for C4.5 decision trees induced with various feature subsets corresponding to the ones presented in Section 5.1.3. The combined CFS subsets comes from applying the CFS algorithm to the complete set of features.

It is evident that the fine-grained PPM sound pressure estimates are more correlated to intensity in music than the coarse-grained RMS estimates (which are only slightly better than chance). The PPM histogram features, which measure dynamics and loudness, are only slightly better than the pure long-term loudness estimates L_{eq}(RLB) and LARM.

We see that the purely spectral features subset scores highest, slightly better than the combined CFS subset, which also includes many spectral features. The increase in accuracy is not large, and since the spectral features subset performs slightly worse on the test data set (Section 5.2.2), we abandon it in favor of the combined CFS subset.

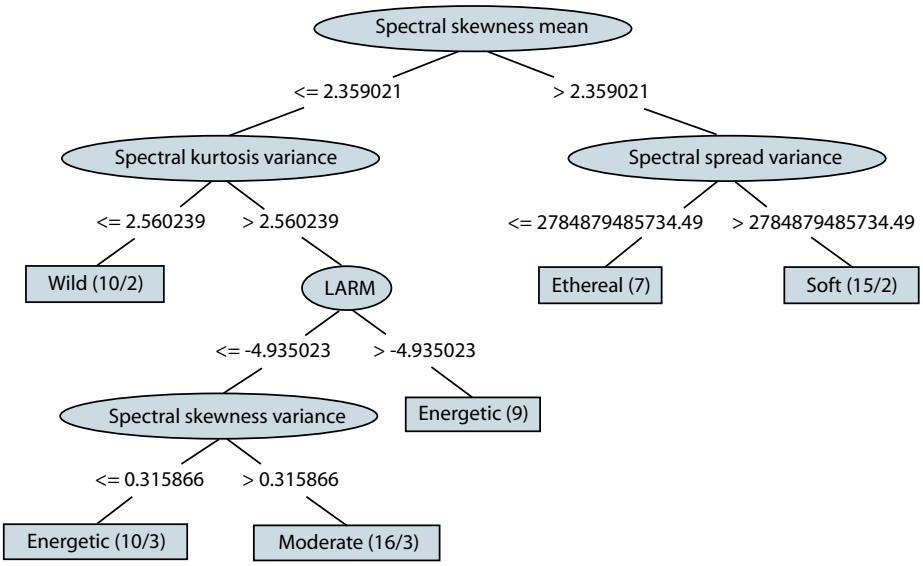


Figure 5.3: C4.5 representation of the intensity model.

5.2.2 Intensity category modelling

A C4.5 decision tree induced with the training data and the optimal feature subset is illustrated in Figure 5.3. The leaf nodes indicate the predicted category label, the total no. of instances and the no. of incorrectly assigned instances of the training data set.

From the figure we see that the final intensity model is based on only five extracted features, possible evidence of good generalization. The average spectral skewness is a strong salient feature, and good indicator of *ethereal* and *soft* vs. the more intense categories. This reinforces the findings of Zils and Pachet [ZP03], where both the spectral skewness and an EDS variation of it showed high correlation to intensity. The *moderate* and *energetic* categories are not as easily separated as the other categories. This is also the only place where long-term loudness estimate (LARM) appears to be a salient feature.

The confusion matrix resulting from 10-fold cross validation over the training data set is shown in Table 5.2. The total accuracy of the model is 62.7%, where 20% is the baseline. (Zils' model, built from 18 features, gave an accuracy of 88.7%.) Because of the relatively small data set, this estimate must be interpreted with some caution. We notice that the instances are located very close to the diagonal. This means that, with exception to only three instances, no instances have been misclassified more than one category in either direction, a kind of inaccuracy that may be tolerable in a practical application.

To evaluate the generalizing capabilities of the final intensity model, we applied it to the test data set, i.e. the music excerpts from the survey that were not accepted into the training data set. Since these did not converge towards a single dominant intensity category, we evaluated the means of the subject ratings to the predicted category labels using Pearson's correlation coefficient r. For the test data set we obtained a strong positive correlation of $r = 0.79$, compared to $r = 0.80$ for the training data itself. This shows that the current intensity model

True	Predicted				
	Ethr	Soft	Modr	Enrg	Wild
Ethereal	**6**	2	1	0	0
Soft	1	**12**	0	0	0
Moderate	0	3	**8**	4	2
Energetic	0	0	8	**10**	2
Wild	0	0	0	2	**6**

Table 5.2: Confusion matrix for the C4.5 intensity model.

is a reliable and reasonably accurate predictor of perceived intensity in music. The intensity model built on purely spectral features obtained a correlation of $r = 0.76$.

5.3 Conclusion and future work

We have presented a descriptive study into perceived intensity of popular music. An intensity taxonomy has been designed for the purpose, and a listening test has been conducted to establish the validity of an objective intensity model. The final intensity model, represented by a decision tree based on low-level audio descriptors, performs reliably for automatic intensity classification.

Intuition tells us that intensity in music is likely to be correlated with mid-level representations such as tempo and "strong beat" (if such), and the effect of adding these to the intensity model will be studied in future studies. Furthermore, we must determine the best way to estimate intensity for complete recordings, not just 20 second excerpts. Taking the median of several segments may be a solution.

Chapter 6

Understanding long-term structure in music

Music structure varies widely from composer to composer and from piece to piece. Transformation, repetition, elaboration and simplification of music materials help to create the unique identity of music. Hence, it is believed that structural description provides a powerful way of interacting with audio content (i.e. browsing, summarizing, retrieving and identifying). Seeing this uniqueness of music structure, it is interesting to ask the question: is it possible to detect non-trivial/significant structural changes (i.e. intro→verse, verse→chorus, chorus→bridge, etc.) in music audio signals?

Here, we present a novel, two-phased approach to this problem based on audio content analysis and similarity computation. In order to obtain appropriate musical content descriptions to detect structural changes, we propose a combination set of low-level descriptors to be extracted from music audio signals. In this chapter, we address the problem of finding acceptable structural boundaries, without prior knowledge about musical structure.

6.1 Computing Structural Descriptions of Music

Audio segmentation facilitates partitioning audio streams into short regions. It seems an indispensable process in certain content-based applications, such as audio notation, audio summarization, audio content analysis, etc. Because of this reason, research in this area has receiving an increasing attention in recent years. A number of different approaches have been proposed [AS01, Foo00, OH04, PBR02, TC99].

Here we propose a novel approach for detecting structural changes in audio signals. We first divide the process in two phases (see diagram in figure 6.1). Each phase is given a different task: Phase 1 focuses in detecting boundaries, which may contain structural changes from the audio signal; Phase 2 focuses in refining detected boundaries obtained from phase 1 by aggregating contiguous segments while keeping those which really mark structural changes in music audio. Our proposed method consists of the following steps:
Phase 1:

1. Segment the input signal into overlapped frames of fixed length and compute audio descriptors for each frame;

2. compute between-frames cosine distance to obtain several similarity matrices [Foo00] for each one of the used features (see section 6.1.1);

3. Apply a morphological filter (see section 6.1.2) to similarity matrices for enhancing the intelligibility of the visualization;

4. Compute novelty measures by applying kernel correlation [Foo00] along the diagonal of the post-processed similarity matrices;

5. Detect segments by finding the first 40 highest local maxima from novelty measure plot;

6. Combine the detected peaks to yield boundary candidates of segment changes of music audio;

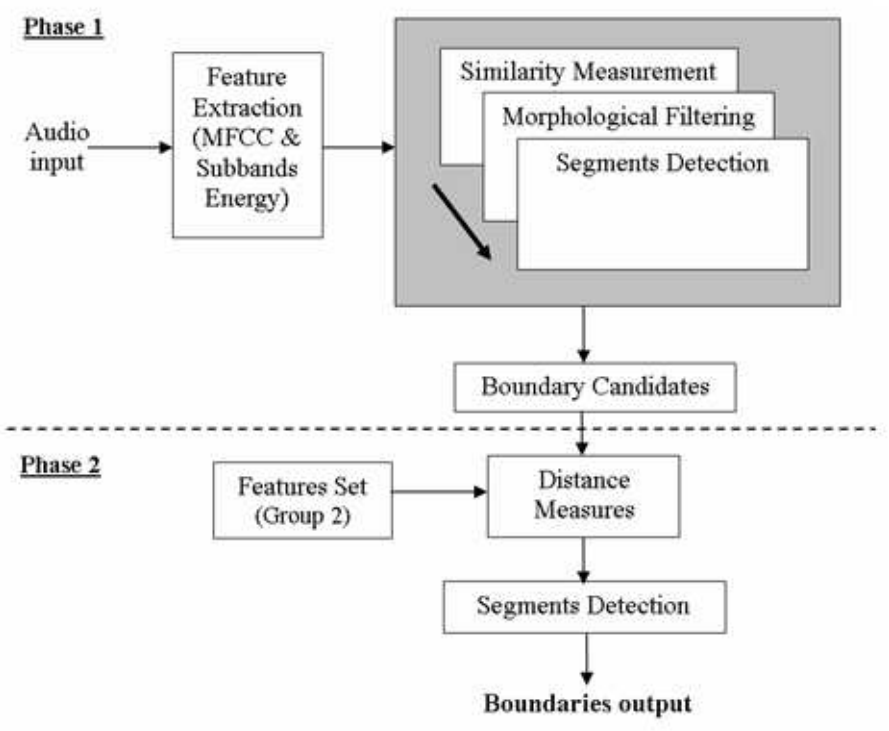


Figure 6.1: Overview framework of our approach

Phase 2:

7. Assign frames according to detected segments obtained from phase 1 and compute the average for all the used features (see table 6.1) in each segment;
8. Compute between-segments' distances using the mean value of each features in each segment;
9. Select significant segments based on a distance measure.

The following sections explain each step in detail.

6.1.1 Feature Extraction

We use a combination of low-level descriptors extracted from music audio signals [Foo99] [Wei03]. The algorithm first segments input signal into overlapped frames (4096-sample window length) with the hop size of 512 samples, then followed by extracting feature descriptions for each of these frames. The proposed features are:

MFCC, also called Mel-Frequency Cepstral Coefficients, a compact representation of an audio spectrum that takes into account the non-linear human perceptual of pitch, as described by the Mel scale.

Phease 1	Phease 2
MFCC Sub-bands Energy	Zero Crossing Rate Spectral Centroid Spectral Flatness Spectral Rolloff Spectral Flux, RMS Low Bass Energy, High-medium Energy

Table 6.1: The list of audio descriptors for Phase 1 and Phase 2.

Spectral Centroid: A representation of the balancing point of the spectral power distribution within a frame.

Sub-bands energy: A measure of power spectrum in each sub-band. We divide the power spectrum into 9 non-overlapping frequency bands as described in [MXKS04].

Zero Crossings: A time-domain measure that gives an approximation of the signal's noisiness.

Spectral Rolloff: A measure of frequency, which is below 95 percentile of the power spectral distribution. It is a measure of the "skewness" of the spectral.

RMS energy: A measure of loudness of the sound frame.

Spectral Flux: The 2-norm of the frame-to-frame spectral magnitude difference vector. It measures spectral difference, thus it characterizes the shape changes of the spectrum.

Spectral Flatness: A measure of the flatness properties of spectrum within a number frequency bands. High deviation from a flat shape might indicate the presence of tonal component.

High-medium energy: A ratio of spectrum content within the frequency range of 1.6 kHz and 4 kHz to the total content. This frequency range comprises all the important harmonics, especially for sung music.

Low Bass energy: A ratio of low frequency component (up to 90 Hz) to the total spectrum content. This frequency range includes the greatest perceptible changes in "bass response".

The computed descriptors are grouped into two sets to be used in different phases of segment detection process. Table 6.1 above shows the grouping of the audio descriptors.

6.1.2 Phase 1 - Rough Segmentation

After computing feature vectors for each frame, we group every 10 frames (116ms) and calculate the mean value for every feature. In this phase of segment detection process, we only work with MFCC and subband energies. We treat those features in separately in order to combine both results in the final stage of detection process in phase 1. In order to find the structural changes in

the audio data, we measure the distance between each feature vectors and their neighbouring vectors using cosine angle distance [Foo00].

To improve the intelligibility of the segment information in the distance representations, we exploit one of the most widely used filtering techniques in image processing field to post-process the computed similarity matrix. We apply a morphological filter [BWFW04] to the similarity matrix to get rid of low value points while keeping the rest of the matrix intact. This is to facilitate the enhancement of the segment boundaries. Following Foote's approaches [Foo00], we then apply a kernel correlation, with the width of 10, along the diagonal of the post-processed similarity matrix to measure the audio novelty. Based on the novelty measure, the first 40 highest local maxima are selected for each individual features. We then aggregate all selected local maxima to yield candidates of segment boundaries for further processing.

6.1.3 Phase 2 - Segment Boundaries Refinement

In the second phase of the detection process, we recombine (join) some of the previously extracted segments into larger units. Here, we consider the within-segment values for all the attributes used in phase 2 (see table 6.1) and compute the segment averages for each of them. Hence, each detected segment now comprises only a set of feature vectors representing the mean value of the attribute in that segment. It has to be mentioned that our used attributes present a large range of values. Apparently, attributes whose values are larger than the other would have more influence in determining the similarity of any two sequences. Hence, in order to avoid such effect and to have an equal importance weight among the used attributes, we normalize all attributes so that its feature values are within the range of 0 and 1. We then compute (dis)similarity between each segment and its neighboring segments by measuring the Euclidean distance between their feature vectors. Similarly to the previous steps in computing novelty measures from the similarity representations, we apply a kernel correlation, along the diagonal of (dis)similarity representation of segments to yield the novelty measures, N, between each segment and its next sequential segment. Finally we select the significant segment boundaries from the computed novelty measures, $N = \{n_s | s = 1, 2, \ldots, l\}$ (where l is the number of segment boundaries candidates) based on the following steps:

1. Select all the peaks that lie above a predefined threshold, Pt, based on their computed novelty measures, N_s , and organize them into a group, which is represented as $P = \{p_i | i = 1, 2, \ldots, M\}$ (M is the number of selected peaks). Whereas those peaks that lie below the predefined threshold, Pt, are organized into another group denoted by $E = \{e_j | j = 1, 2, \ldots, Q\}$($Q$ is the number of unselected peaks).

2. Organize all peaks in E in ascending order according to their distance measures.

3. Select the highest peak in E for further evaluation.

4. Based on temporal information, if the evaluated peak is located at least 4 sec apart from any peaks in P, insert it in group P and reorganize all peaks in group P in ascending order based on the segment index number;

otherwise delete it from E. This is based on the assumption that each section in music (ex. verse, chorus, etc.) should at least hold 4 sec (1 bar for songs with quadruple meter with 60 bpm tempo) in length before moving to the next section.

5. Go to step 3.

The whole iterative peak selection process ends when there is no more peak in E. Finally, segment boundaries in P are considered as significant segment boundaries that mark structural changes in music audio signals.

6.2 Results

6.2.1 Data Set

In our experiments, we use the 54 songs from first four CD's of The Beatles (1962 - 1965) as a test set. Each song is sampled at 44.1 kHz, 16-bit mono. For algorithm evaluation, we have generated a ground truth by manually labelling all the sections (i.e. intro, verse, chorus, bridge , verse, outro, etc.) of all the songs in the test set, according to the information provided by Allan W. Pollack's "Notes On" Series website on song analyses of Beatles' twelve recording project . A music composer supervised the labelling process and results.

6.2.2 Recall and Precision Measures

To quantitatively evaluate the detected segments from the proposed algorithm against the ground truth, we calculate the precision and recall and F-measure measures of the test. In evaluating the identified segment, we allow a tolerance deviation of ± 3 seconds (approximately 1 bar for a song of quadruple meter with 80 bpm in tempo) from the manually labelled boundaries. Precision and recall are mainly used to evaluate the accuracy and reliability of the proposed algorithm, whereas F-measure is mainly used to measure overall effectiveness of detection by combining recall and precision with an equal weight.

6.2.3 Results

Our proposed structural changes detection approach has achieved accuracy higher than 71% and a reliability of 79% using the ground truth set. The overall F-measure reached 75%. In another words, with 10 detected segment boundaries, 7 of them are correctly detected compared to ground truth data. Whilst about 2 out of 10 manually labelled boundaries are missed by our automatic boundaries detector. The distribution of precision scores has a standard deviation of 0.11 and the range of precision values spans across 0.41-0.94. From our results, we observe that the best performance in the case of SongID-35 with its recall and precision score of 100% and 94% respectively. Whereas the worst performance is observed in the case of SongID-47, which only reaches the recall rate of 38% and precision rate of 56%. Figure 6.2 illustrates the detected segment boundaries by our proposed algorithm with manually labelled segment boundaries for SongID-35. It is worthwhile to pay attention to the fact that precision and recall rate are particularly low for those songs which comprise

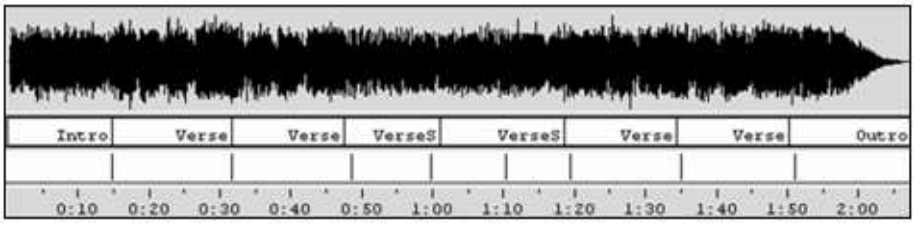


Figure 6.2: Manually labelled segment boundaries (top) and detected segment boundaries by our proposed algorithm (middle) with time position information (below) for SongID-35 entitled "Words of Love". The label VerseS means an instrumental solo playing the verse. Labels are not yet assigned by the algorithm.

of smooth transition between sections. It seems that our descriptors are not sensitive enough to mark these changes. On the other hand, songs with coarse transition between sections usually achieve a better rate on these measures. Using some other disregarded descriptors perhaps may be able to cope with this matter.

With our test data set, we have compared our approach with previous method described in [Wei03]. It was reported that with an allowed tolerance deviation of 3.7 sec (higher than ours), the author reported less than 60% for both recall and precision measures respectively whereas we achieved over 70% for these both measures. However, it should be noted that the generality of our music database is quite limited. So far, we have not yet tested our approach on different music genres (i.e., instrumental music, techno or jazz).

6.3 Conclusions

We have presented a new approach for detecting structural changes in music audio using a two-phased procedure and different descriptors for each phase. A combination set of audio descriptors has also been shown useful in detecting music structure changes. Evaluation results have shown the validity and the performance of our proposed approach. For ongoing research to further improve the detection algorithm, more attention will be given to the following factors: Integration of some previously disregarded lower-level feature attributes (i.e. Pitch class profile, etc.); Automatic labelling of sections according to their structural title (i.e. intro, verse, bridge, etc.); Testing the performance using an annotated database comprising different music genres different from "60's pop music" (for which we plan to use the approach in Appendix C); and most importantly in the context of SIMAC, Making use of higher-level analysis techniques (i.e. beat detection, phrase detection, etc.) to achieve better segment truncation.

In this later direction we will present an example of long-term segmentation using the harmonic mid-level representation in Chapter 3.

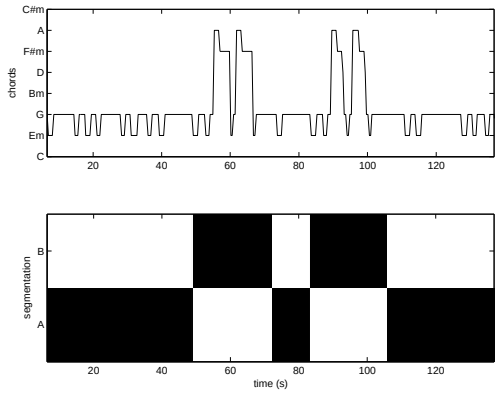


Figure 6.3: *Love me do* by The Beatles: estimated chord sequence (top) and estimated segments, showing the long-term structure "ABABA".

6.4 Future work: using high-level descriptors for segmentation

In this section we demonstrate, in principle, how long-term segmentation of songs can be achieved using higher-level semantic description. As an example we perform the following experiment using the harmonic representation in Section 3.3 and a histogram clustering algorithm developed by [ACN+05]. The algorithm calculates a sequence of unlabelled states (e.g. A and B) that represent the long-term sections of a song (e.g. chorus, verse, bridge, etc) from a sequence of histograms computed from our labelled sequence. It consists of a phase of simulated annealing to learn the state transition probability matrix [[PBH99]] and a second phase of combined annealing and Gibbs sampling to compute the posterior probabilities of segments belonging to given states, and thus the sequence of states (see [RC99] for an introduction).

The top plot of Fig. 6.3 shows the resulting chord labelling for *Love me do*, the song on which the harmonic labelling algorithm performed the worst. The bottom plot shows, for each time step, the marginal posterior probabilities obtained from the segmentation algorithm, such that white indicates zero probability and black indicates a probability of 1. From both these plots we can clearly see the simple structure of the song, of the form "ABABA". This demonstrates how, even when imperfect, our high-level representation is consistent, allowing for successful clustering of its symbols. To our knowledge, this success is the first example of long-term segmentation using a mid-level harmonic feature set.

Figure 6.4 shows segmentation results for a more complicated structure, that of "Wonderwall" by Oasis. The top plot shows the calculated sequence of chord labels ("chords"). The next line ("true") shows the manually annotated segments of the song. The middle line depicts the automatically segmented sections using the chord labels ("seg1"). Finally, the bottom line ("seg2") shows the automatically segmented sections obtained after first collapsing our tactus-based chord labels (e.g. CCGGFFFEAAAA) into a simple sequence of chords

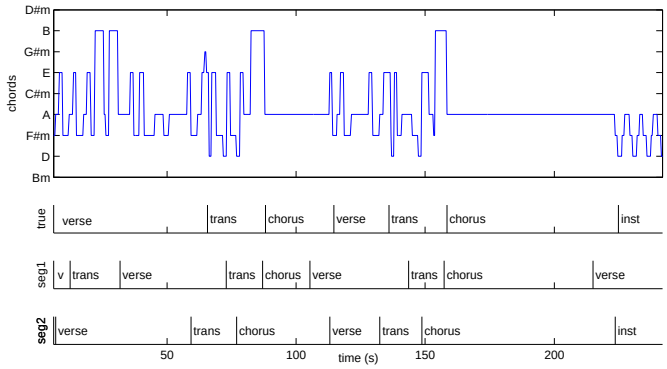


Figure 6.4: Estimated chord sequence (top) and long-term segment boundaries from *Wonderwall* by Oasis: "true" refers to ground-truth annotation, "seg1" to segments obtained using the heamonic representation and "seg2" to segments obtained by shrinking the representation into a simple chord sequence.

(e.g. CGFEA) by removing contiguous duplicates.

As can be seen in "seg1", there are some problems with the segmentation: the verse is segmented as to include parts of the transition, the chorus section and a final instrumental *Coda*, creating some confusion between them, and thus resulting in mislabelling. On the other hand, segmentation on the collapsed chord sequence is more accurate, both in terms of temporal localization and segregation between states. We suggest that this is because the resulting chord groupings can be thought of as equivalent to musical phrases. Indeed, some informal testing seems to support the idea that when the number of segmentation states is increased and the length of our histograms is reduced, we start to pick up segments that are related to sections at a shorter temporal scale (e.g. phrases). While a proper study on segmentation is currently in progress, we suggest that this increased granularity is potentially a major asset of harmonic-based segmentation, in opposition to segmentation based on low-level features, where short-term structures are not necessarily indicative of musical gestures.

Chapter 7

Music Complexity within songs

7.1 Background and Motivation

Music complexity in this context is understood as a multifaceted, semantic descriptor of musical audio content. We regard the complexity of music as a high-level, intuitive attribute, which can be experienced directly or indirectly by the active listener. In particular we define the complexity as that property of a musical unit which determines how much effort the listener has to put into following and understanding it.

In the context of SIMAC it is the goal to provide operational models for the automated computation of music complexity as it is perceived by human listeners. We are aiming here at an estimation on a rough scale, that does not assume any particular musical expertise and addresses the "common sense"-idea of music complexity. The focus are music and listeners from a western cultural background.

Since music has different and partly independent facets, we believe these should be addressed individually with separate models. The following musical facets can be considered:

- Rhythm
- Harmony and Melody
- Timbre
- Acoustics (spatial/dynamic)
- Structure

As with the other descriptors presented in this document we believe in the possibility of enhanced interaction with music collections by making information about music complexity available. Among the obvious application for visualization, clustering or explicit querying, a complexity description has other interesting properties that can be useful in this context. One advantage is the very compact representation, since each music track will be assigned only one complexity value for each of the considered facets. But more interesting is the relationship between complexity and preference. This has been put forward in the theory of Arousal Potential by Daniel Berlyne already in 1971 [Ber71]. In this publication he states that an individual's preference for a certain piece of music is related to the amount of activity it produces in the listener's brain, to which he refers as the *arousal potential*. According to this theory there is an optimal arousal potential that causes the maximum liking, while a too low as well as a too high arousal potential result in a decrease of liking. He illustrates this behaviour by an inverted U-shaped curve (see figure 7.1) which was originally introduced in the 19th century already by Wundt to display the interrelation between pleasure and stimulus intensity. Since then many experiments have been conducted showing clear interdepence between the complexity of musical instances and the preference for them (either in individuals [Hey75], [NH97], or measured by popularity [EN00], [Par04]). So we can assume that complexity might be of relevance also in music recommendation.

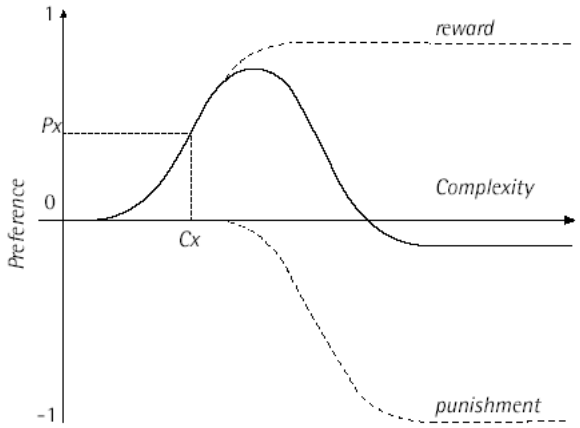


Figure 7.1: The Wundt curve for the relation between music complexity and preference.

7.2 Implementations

In this section two implementations of complexity descriptors are explained in detail. Both have been programmed in MATLAB and were tested on a large music collection. The evaluation results are reported in section 7.3 starting on page 138.

7.2.1 Dynamic Complexity

Considering the dynamic complexity (which we can understand as one component of the acoustic facet of complexity) the computation of instantaneous loudness is a key issue. The implementation described here uses for efficiency reasons a simplified computational model that was described by Earl Vickers in [Vic01]. Some modifications and additions have been made in order to make the algorithm fit for the desired task.

As a first step the algorithm applies a very simplified "B" weighting function to the audio signal in order to account for the human ear's transfer function. For efficiency reasons this weighting is done by a first-order Butterworth high-pass filter with a cut-off frequency of 200 Hz. So actually only the low end of the weighting function is approximated. In the case of music signals this is tolerable, since they have much more energy in the bass and mid-range compared to the high frequencies.

The pre-emphasized signal $x_{pre}(n)$ is then fed into a root-mean-square level detector with an integrated smoothing function. This level detector consists of a running average V_{ms} (eq. 7.1) that is downsampled according to the chosen frame length N.

$$\begin{aligned} V_{ms}(n) &= c \cdot V_{ms}(n-1) + (1-c) \cdot x_{pre}^2(n) \quad \text{, with} \\ c &= e^{-\frac{1}{\tau F_s}} \end{aligned} \tag{7.1}$$

F_s corresponds to the sampling frequency and τ is the time constant for the smoothing (35 ms in this implementation). The downsampling is done according

to equation 7.2. We chose framesize of 200 ms corresponding to $N = 8820$ for a sampling rate of 44.1 kHz.

$$V_{rms}(i) = \sqrt{V_{ms}(N \cdot i + N - 1)} \tag{7.2}$$

The instantaneous level is then converted to dB by calculating

$$V_{dB} = 20 \cdot \log_{10}\left(V_{rms}(i)\right). \tag{7.3}$$

In order to avoid silence at the end or the beginning of the track to have an effect on the complexity estimation, successive frames with a level below -90 dB are deleted when they appear at either end. Afterwards, the global loudness level L according to Vickers is calculated. L is a weighted average of all M instantaneous level estimates, where the louder ones are assigned a higher weight:

$$\begin{aligned} L &= \sum_{i=0}^{M-1} w(i) \cdot V_{dB}(i) \quad \text{, with} \\ w(i) &= \frac{u(i)}{\sum_{j=0}^{M-1} u(j)} \quad \text{and} \\ u(j) &= 0.9^{-V_{dB}(j)} \end{aligned} \tag{7.4}$$

The emphasis on the loud frames is grounded in psychoacoustic findings. So for example Zwicker and Fastl [ZF90] suggest that the loudness of a dynamically changing sound can be characterized by the loudness level which only 5% of the frames exceed. In this implementation a variation of Vickers *dynamic spread* is used as the final dynamic complexity measure. It is simplified in the sense that periodic loudness variation and the suddenness of changes are not considered. Instead it comprises basically the mean distance from the global loudness level (eq. 7.5) so that high values correspond to higher complexity and vice versa.

$$C_{dyn} = \frac{1}{M} \sum_{i=0}^{M-1} |V_{dB}(i) - L| \tag{7.5}$$

7.2.2 Rhythmic complexity: danceability

Danceability (i.e. a measure of how easy it is to dance to the music) is part of the rhythmic complexity facet. The implementation presented here is following the descriptions by Jennings et al. [JIM+04]. Where exact specifications were missing the reasonable solutions were integrated.

As a first step the audio signal is segmented into non-overlapping blocks of 10 ms length. For each block the standard deviation $s(n)$ of the amplitude is computed. The values $s(n)$ resemble a bounded, non-stationary time series, which can be associated with the averaged physical intensity of the audio signal in each block (see figure 7.2). In order to obtain the unbounded time series $y(m)$, $s(n)$ is integrated:

$$y(m) = \sum_{n=1}^{m} s(n) \tag{7.6}$$

This integration step is crucial in the process of DFA computation, because for bounded time series the DFA exponent (our final feature) would always be 0

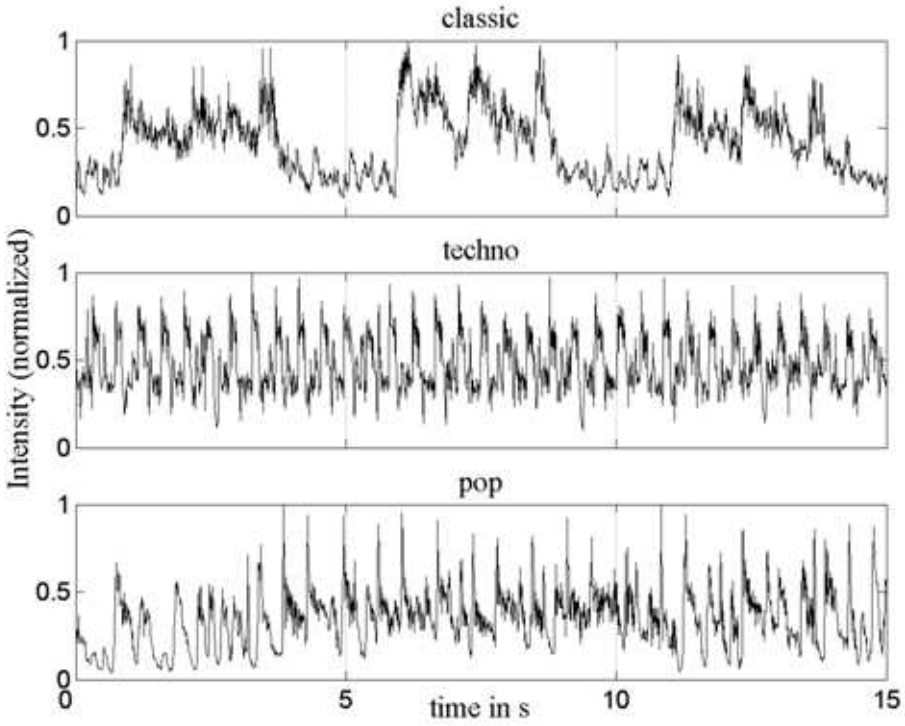


Figure 7.2: Excerpts from the time series $s(n)$ for three example pieces from different musical genres.

when time scales of greater size are considered. This effect is explained in more detail in [Pen].

The series $y(m)$ can be thought of as a random walk in one dimension. $y(m)$ is now again segmented into blocks of τ elements length. This time, we advance only by one sample from one block to the next in the manner of a sliding window. There are two reasons for this extreme overlap. First, we obtain more blocks from the signal, which is of interest, since we will obtain better statistics from a larger number of blocks. Secondly, we avoid possible synchronization with the rhythmical structure of the audio signal, which would lead to arbitrary results depending on the offset we happen to have. However, performing the computation in this manner the number of operations is enormously increased.

From each block we now remove the linear trend $\hat{y}_k$ and compute $D(k, \tau)$, the mean of the squared residual:

$$D(k,\tau) = \frac{1}{\tau} \sum_{m=0}^{\tau-1} \left(y(k+m) - \hat{y}_k(m)\right)^2 \tag{7.7}$$

We then obtain the detrended fluctuation $F(\tau)$ of the time series by computing the square root of the mean of $D(k, \tau)$ for all K blocks:

$$F(\tau) = \sqrt{\frac{1}{K} \sum_{k=1}^{K} D(k,\tau)} \tag{7.8}$$

As indicated, the fluctuation F is a function of τ (i.e. of the time scale in focus). The goal of DFA is to reveal correlation properties on different time scales. We therefore repeat the process above for different values of τ that are within the range of our interest. Jennings et al. [JIM+04] use a range from 310 ms ($\tau = 31$) to 10 s not specifying the step size in their paper. Relating these time scales to the musical signal they are reaching from the beat level through the bar level up to a level of simple rhythm patterns.

The DFA exponent α is defined as the slope on a double log graph of F over τ (eq. 7.9). It therefore makes sense to increase τ by a constant multiplication factor rather than a fixed step size. Apart from giving equally spaced supporting points on the logarithmic axis it also reduces the computational operations without affecting the accuracy too much. We chose a factor of 1.1 giving us 36 different values for τ covering time scales from 310 ms to 8.8 s.

For small values of τ an adjustment is needed in the denominator when computing α (cp. [BGH+95]) giving us the following formula for the DFA exponent:

$$\alpha(i) = \frac{\log_{10}\left(F(\tau_{i+1})/F(\tau_i)\right)}{\log_{10}\left((\tau_{i+1}+3)/(\tau_i+3)\right)} \tag{7.9}$$

As τ grows, the influence of the correction becomes negligible. In case that the time series has stable fractal scaling properties within the examined range, the double log graph of F over τ is a straight line making $\alpha(i)$ a constant function. We find a constant value of 0.5 for a completely random series (white noise), a value of 1 for a series with $1/f$-type noise, and 1.5 for a Brown noise series (integrated white noise) [Pen].

For music signals normally we don't have stable scaling properties (cp. figure 7.3). Opposed to heart rate time series for example, there is much more variance

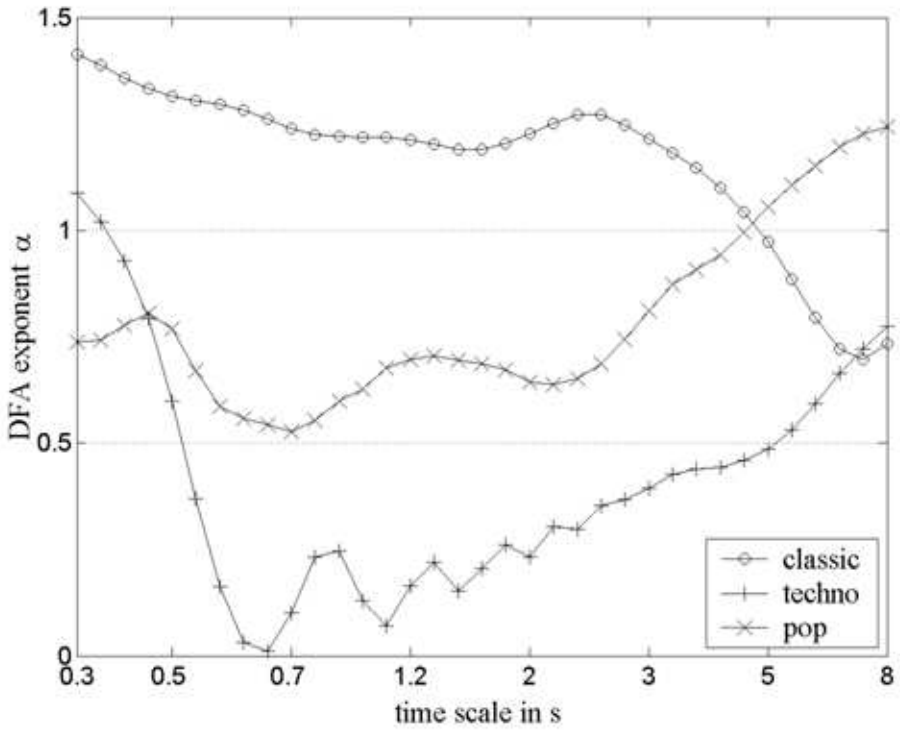


Figure 7.3: DFA exponent functions for the three example tracks from figure 7.2.

in $\alpha(i)$ for music. Still, we can find that music with sudden jumps in intensity generally yielding a lower level of $\alpha(i)$ than music with a smoother varying series of intensity values. That means music with pronounced percussion events and emphasized note onsets shows lower α values than music with a more floating, steady nature. Apart from that, there is also a relationship between strong periodic trends and the α function. A more detailed discussion on this can be found in [SH05].

In order to arrive at an indicator for the danceability and thus a certain aspect of the rhythmic complexity the α values have to be further reduced. While different ways are thinkable to do this, in this implementation simply the average α level was computed for each track. A high value refers to a high complexity (not danceable), a low value refers to a low complexity (highly danceable).

7.3 Evaluation

The evaluation of complexity descriptors is not easy. The ideal condition for evaluation is of course a solid ground truth annotation against which the performance of the extraction algorithms can be measured. Unfortunately these ground truth annotations are rather troublesome to come by. Especially for complexity we have the problem that, other than for genre or tempo for example,

available annotations usually do not cover this issue. The manual annotation of the own testing material needs a lot of time and resources and might not be feasible in some cases.

The two descriptor implementations described above were evaluated by subjective means on the basis of a large music collection. This was done by randomly picking tracks at different complexity levels and judging the danceability and the dynamic complexity in direct comparison by listening. A formal user study was not carried out.

For the danceability descriptor also a more objective evaluation was done. General statistical methods and machine learning methods were applied in order to explore relations between the semantic labels, or certain artists and the DFA exponent. The rationale behind this is to prove a systematic variation of the DFA exponent subject to certain semantic attributes assigned to the music. We will only shortly report these results here, more details can be found in [SH05].

7.3.1 The Dataset

The two descriptor implementations described above were computed on a large music collection. A data set of 7750 tracks from MTG-DB [CKF+04a], a digital music collection from the MTG lab, was used in the experiment. Each track refers to a full piece of music. The dataset also contained annotated semantic labels for each item, which were obtained from music experts and afficionados, and had been manually assigned to the tracks. In our experiments we used the artist names and also "tone" labels consisting in abstract attributes that are associated with the music, such as "Rousing", "Sentimental", or "Theatrical". The list of "tone" labels is composed of a total of 172 different entries. In the statistical analysis only a subset of 136 labels were considered, because the remaining ones appeared less than 100 times each. It must be noted that these labels are originally assigned to the artists and not to the individual tracks. Therefore a certain degree of fuzziness has to be accepted with these descriptions when working on the track level. The data set contained very different, mostly popular styles of music from a total of 289 different artists. A maximum of 34 labels were assigned to a single artist, while the average was 11 labels per artists. Figure 7.4 shows a bar plot of the eight labels that were assigned to the highest number of artists. The average number of artists sharing a label was 18.

7.3.2 Results

By the manual random checks it was found that the complexity estimations at the extreme ends were the most consistent ones. Comparing the tracks from these regions with each other and with the intermediate ones the underlying concept of the descriptors immediately became apparent. This effect can be easily seen in figure 7.5, where the 60 highly danceable techno music tracks can be almost perfectly separated from the 60 non-danceable film score tracks only by considering their average α value. The fine grain ranking within a local region however didn't appear comprehensible in many cases. This was especially noticeable in the very dense area of intermediate values. So rather a coarse classification into 3–5 complexity levels than a continuous ordering of the entire collection was achieved.

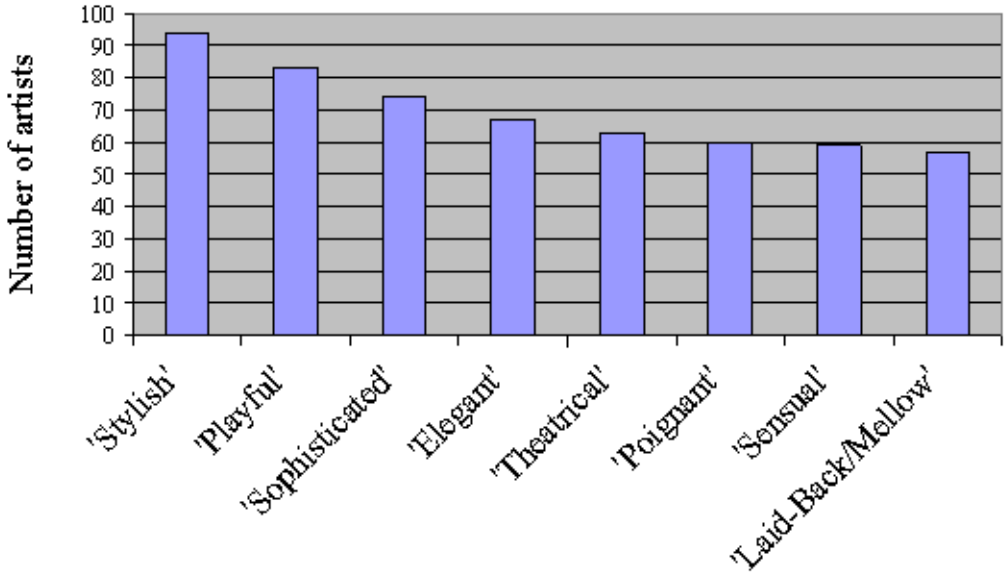


Figure 7.4: Top-eight labels with the highest number of assigned artists.

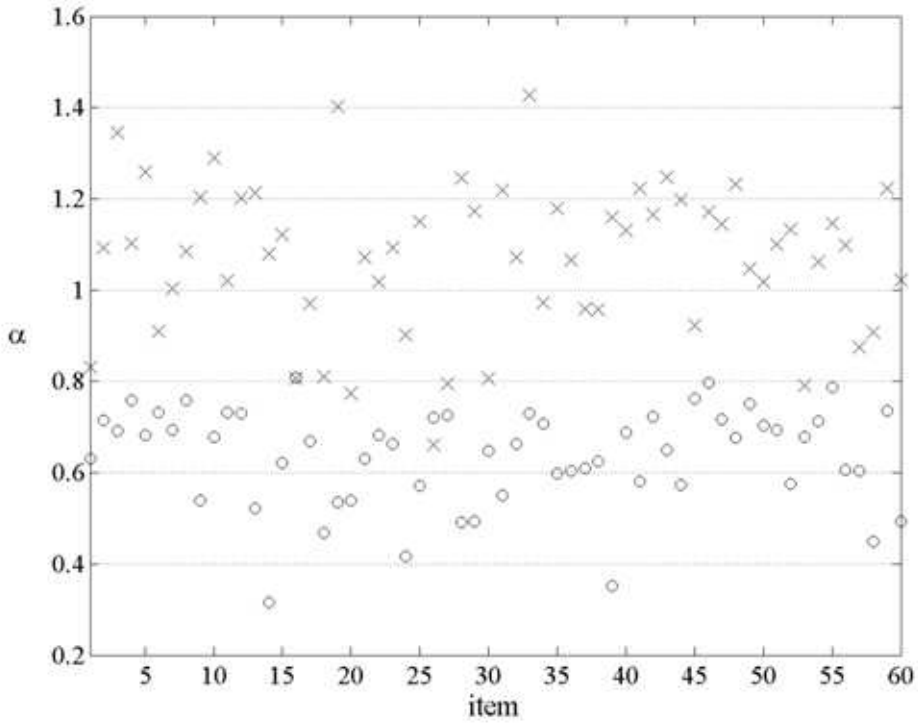


Figure 7.5: α-levels for 60 techno (o) and 60 film score tracks (x), unordered.

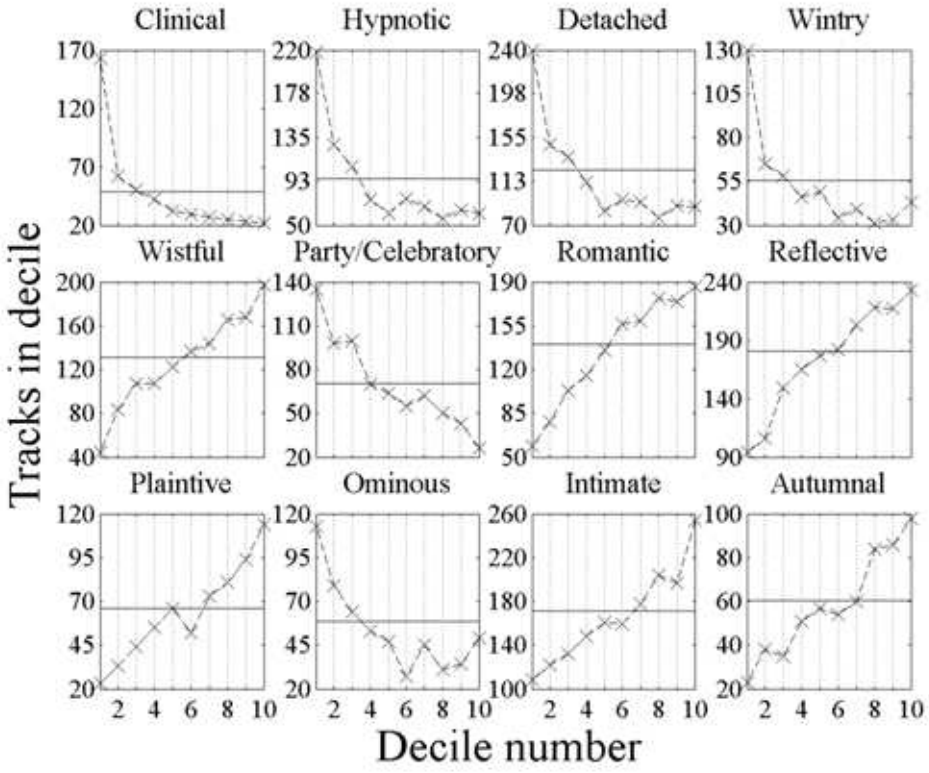


Figure 7.6: Distributions on deciles for the twelve labels with most significant deviation from equal distribution (solid lines).

The results of the statistical tests for the danceability descriptor sustain the findings from manual random evaluation. Strong coherence of high statistical significance was found for several of the "tone" labels that are semantically close to the concept "danceable" or "not danceable" respectively. For example the labels "Party/Celebratory" and "Energetic" in the context of music have a clear relation with danceability, whereas "Plaintive" and "Reflective" appear more appropriate descriptions for music that is not well suited for dancing.

The results reveal a consistency on a high abstraction level even exceeding the aspect of danceability. Figure 7.6 shows how the distribution of some labels on the deciles starting from the lowest to the highest α values in the collection. The distribution of α values on the whole collection was normal with a mean of 0.863. The tracks assigned to each label were tested for significant deviations from this distribution. Only those with normal distributions were considered. Of these, 24 showed a significantly higher and 35 a significantly lower mean value. When looking at the two lists of labels a certain affinity can be noted in many cases on either side. The group of labels for higher α wakes associations of *Softness*, *Warmness*, *Tranquility*, and *Melancholy*. For the others we might form two subgroups, one around terms like *Exuberance* and *Vehemence*, the other around *Tedium* and *Coldness*. Comparing labels from both lists with each other, we can identify several almost antonymous pairs, for ex-

ample: Outrageous - Refined/Mannered, Boisterous - Calm/Peacefull, Carefree - Melancholic, Clinical - Intimate.

In the machine learning experiments two artist classification tasks where tried. It must be stated again here, that the "tone" labels mentioned above originally also belong to the artists and thus only indirectly to the individual tracks. In both experiments the classification accuracy was clearly above chance level with 73% over 50% and 67% over 48% respectively proving also the semantic significance of the descriptor.

7.4 Conclusions

In this chapter we have presented one of the novel aspects of our semantic description scheme, that of music complexity. Complexity can be calculated in all the different musical dimensions of our descriptions scheme, as presented in previous chapters, and is often closely associated to attributes that users search for in music retrieval systems. Two example implementations, for dynamic and rhythmic complexity, have been shown and evaluated. These illustrate the validity of the concept and its potential for semantic description of music contents.

Chapter 8

Music Similarity between songs

Within SIMAC different types of similarity have been investigated. The primary focus has been on audio-based similarity. That is, using only information which can be extracted directly from the audio signal. In addition, in WP3 (D3.1.1, D3.2.1, D3.4.1) computing similarity based on information extracted from web pages has been investigated.

This chapter discusses only the audio-based similarity developed within SIMAC. In particular, similarity measures as used for genre classification, playlist generation, or recommendation (which are functionalities of the SIMAC organizer and recommender prototypes) are presented.

Parts of this chapter are covered in more detail in the WP3 Music Similarity Report D3.4.1. While the development of audio-based descriptors has been the core of WP2, the application and evaluation of these for genre classification and playlist generation has been the core of WP3.

This chapter is structured as follows. First we briefly discuss the evaluation of similarity. The way we evaluate the similarity is perhaps the best way to define the type of similarity we are considering. Second, spectral similarity is reviewed. Third, fluctuation patterns are described together with two descriptors extracted thereof. Fourth, we review experiments aiming at combining fluctuation patterns and spectral similarity. Fifth, preliminary work aiming at overcoming the glass ceiling which we encountered using low-level audio statistics is presented. Finally, we summarize and discuss this chapter.

8.1 Evaluation

Music similarity is a multi-dimensional metric. Two pieces can be similar because they have similar lyrics, instrumentation, chord progressions, melodies, rhythm, moods, etc. In this chapter we define similarity as the concept which songs within a genre (or subgenre) have in common (see also D3.4.1). As already pointed out in D3.1.1 and D3.2.1, the reason for using such a simplistic definition, is because it allows highly efficient (i.e. fast and cheap) evaluations of the similarity measures (since genre labels for artists are readily available). In particular, we primarily use nearest neighbor classifiers (and genre classification) to evaluate the similarity measures. We assume that pieces within the same genre are very close to each other.

8.2 Spectral Similarity

The basic idea of spectral similarity is to compare the spectra of two songs with each other. This is usually done without taking time information into account. (I.e. playing a song forwards of backwards would produce the same results. See D3.4.1 for a discussion on this.) There are different ways to compute spectral similarity (see D3.2.1). In particularly, useful we have found the approach presented by Aucouturier and Pachet (AP) [AP02b]. This approach models the distribution of spectra within a song with a Gaussian mixture model. We refer to this model as the "cluster model". The similarity between two models is computed by sampling from one model and computing the likelihood of this sample in the other model (Monte Carlo sampling).

The main problem with spectral similarity as suggested by AP is that it

is very slow. Analyzing a song and computing the Gaussian mixture model might take 1 minute. Comparing two songs might take 1 second. Reducing the computation time (and thereby developing new algorithms) is ongoing work within SIMAC. Preliminary results indicate that a speed-up of a factor up to 100 might be possible.

8.2.1 Details

For the experiments reported in the WP3 deliverables (D3.1.1, D3.2.1, D3.4.1) we used the implementations in the MA Toolbox [Pam04] and the Netlab Toolbox[1] for Matlab.

From the 22050Hz mono audio signals two minutes from the center are used for further analysis. The signal is chopped into frames with a length of 512 samples (about 23ms) with 50% overlap. The average energy of each frame's spectrum is subtracted. The 40 Mel frequency bands (in the range of 20Hz to 16kHz) are represented by the first 20 MFCC coefficients. For clustering we use a Gaussian Mixture Model with 30 clusters and trained using expectation maximization (after k-means initialization). The cluster model similarity is computed with Monte Carlo sampling and a sample size of 2000.

8.3 Fluctuation Patterns

Fluctuation Patterns (FPs) describe loudness fluctuations per frequency bands [Pam01, PRM02]. They describe characteristics of the audio signal which are not described by the spectral similarity measure.

First, the audio signal is cut into 6-second sequences. The center 2 minutes from each piece of music are used and cut it into non-overlapping sequences. For each of these sequences a psychoacoustic spectrogram, namely the Sonogram is computed. For the loudness curve in each frequency band a FFT is applied to describe the amplitude modulation of the loudness.

From the FPs two new descriptors are extracted. The first one, describes how distinctive the fluctuations at specific frequencies are, it is called *Focus*. The second one is called *Gravity* and is related to the overall perceived tempo. Details on these descriptors can be found in [PFW05].

8.3.1 Details

For the experiments reported in WP3 the MA toolbox was used. The number of Bark bands for the sonogram was set to 20 and the FFT hopsize to 50% of the FFT window size which is 23ms. A "hann" window function was used. The dimensionality of the fluctuation patterns is 1200 (60 modulation frequencies from 0 to 10Hz and 20 Bark bands). Compared to the AP spectral similarity the computation time is negligible. The main advantage is that the FP define a vector space. Within this space two songs can be compared very quickly using the Euclidean distance. However, experiments have also shown that AP performs better than the fluctuation patterns.

[1] http://www.ncrg.aston.ac.uk/netlab

8.4 Combination

A very straightforward idea is to combine a well working similarity measure such as the one suggested by Aucouturier and Pachet with complementary similarity measures such as the fluctuation patterns.

To combine the spectral similarity with the fluctuation patterns and the two descriptors we use a linear combination similar to the idea used for the aligned Self-Organizing Maps (SOMs) [PGW03]. Before combining the distances we normalize the four distances such that the standard deviation of all pairwise distances within a music collection each equals 1. In contrast to the aligned-SOMs we do not rely on the user to set the optimum weights for the linear combination, instead we automatically optimize the weights for genre classification.

The details of the experiments are described in D3.4.1. Basically, we have found that 65% AP combined with 15% fluctuation patterns, 5% focus, and 15% gravity performs best. Any new descriptors developed in WP2 can be tested if including them into a similarity measure will increase performance. Of particular interest are descriptors related to harmonic information or rhythmic information as both dimension of similarity are currently not covered by the combination suggested in this section.

8.5 Chroma-Complexity Similarity

In this section we review some of the results presented in D3.4.1 and recent work building on higher level representations. The goal is to develop semantic descriptors which will be used in a combined similarity measure. The prototypes could benefit from similarity measures reflecting higher level musical concepts. These might make the prototypes more interesting for people with specific interests in music. For example, pieces with a exceptionally strong deviation in the chroma complexity could be removed from a playlist. Another example would be to organize a music collection according to chroma complexity. For example, to group all pieces with a very high complexity.

In the current state the work is very exploratory with the goal to gather different ideas of how mid and high level descriptors developed in WP2 can best be applied to specific WP3 tasks such as genre classification or playlist generation. Thus, we heavily rely on interactive visualization tools for better understanding of the data and different parameters involved. In this section we will describe three of the tools we developed for this purpose.

The computation and the tuning of the chromagram are described in detail in Section 3.2. In this section we only describe the processing based on a pre-computed chromagram. Further work will include using higher level representations as described in Section 3.3.

8.5.1 Chromagram Processing

To emphasize certain patterns in the chromagram and to remove temporal variations we use several filters. (1) We use a Gaussian filter over time. This window is very large and removes variations within 50ms. This helps reduce the impact of, for example, the broad spectrum of sharp attacks. (2) We use a loudness

normalization to remove the impact of the changing loudness levels in different sections of the piece. (3) We use gradient filters to emphasize horizontal lines in the chromagram. (4) We smooth the chromagram over the tone scale to a resolution of about one semitone (i.e. 12 bins instead of 36). This smoothing is done circular in accordance with the distance between semitones. The result are the chromagrams displayed in the figures of this chapter.

8.5.2 Chroma Complexity

Depending on the number of chords and their similarity, the patterns which appear in a chromagram might be very complex, or very simple. To measure this we use clustering. In particular, the chromagram is clustered with k-means, finding groups of similar chroma patterns. The cluster algorithm starts with 8 clusters. If two clusters are very similar, the groups are merged. This is repeated until convergence or until only 2 groups are left. The similarity is measured using a heuristic for perceptual similarity of two patterns. To avoid getting stuck in local minima, the clustering is repeated several times (with different initializations). (The time resolution of the chromagram is very low, thus the computation time for clustering is negligible.) This is not the optimal choice for several reasons. Alternatives include using, for example, the Bayes information criteria, or avoiding quantization in the first place.

In the following we will give some examples to illustrate the chroma complexity and to show that there are general tendencies for genres which make the approach interesting for tasks such as genre classification or playlist generation.

8.5.3 ChromaVisu Tool

To study the chromagram patterns we developed a Matlab tool to visualize patterns while listening to the corresponding music. A screenshot is shown in Figure 8.1. The six main components are:

A. The large circle in the upper left is the current chroma-pattern (i.e. the pattern associated with the part of the song just playing).

B. To the right is the mean over all patterns which for many types of music is a useful indicator of the key.

C. To the right are up to eight different chroma patterns which occur frequently. The number above each pattern indicates how often it occurs (percentage). The number of different patterns is determined automatically as described above and is the measure for the chroma complexity.

D. Beneath is a fuzzy segmentation (cluster assignment) indicating when which of the eight patterns is active. Each line represents one cluster (with the most frequent cluster in the first row). White means that the cluster is a good match, black is a very poor match. If none of the clusters is a good match, the last line (which does not represent a cluster) is white. Modeling the repetitions and the structure which immediately become apparent are the primary target for further work on chroma complexity.

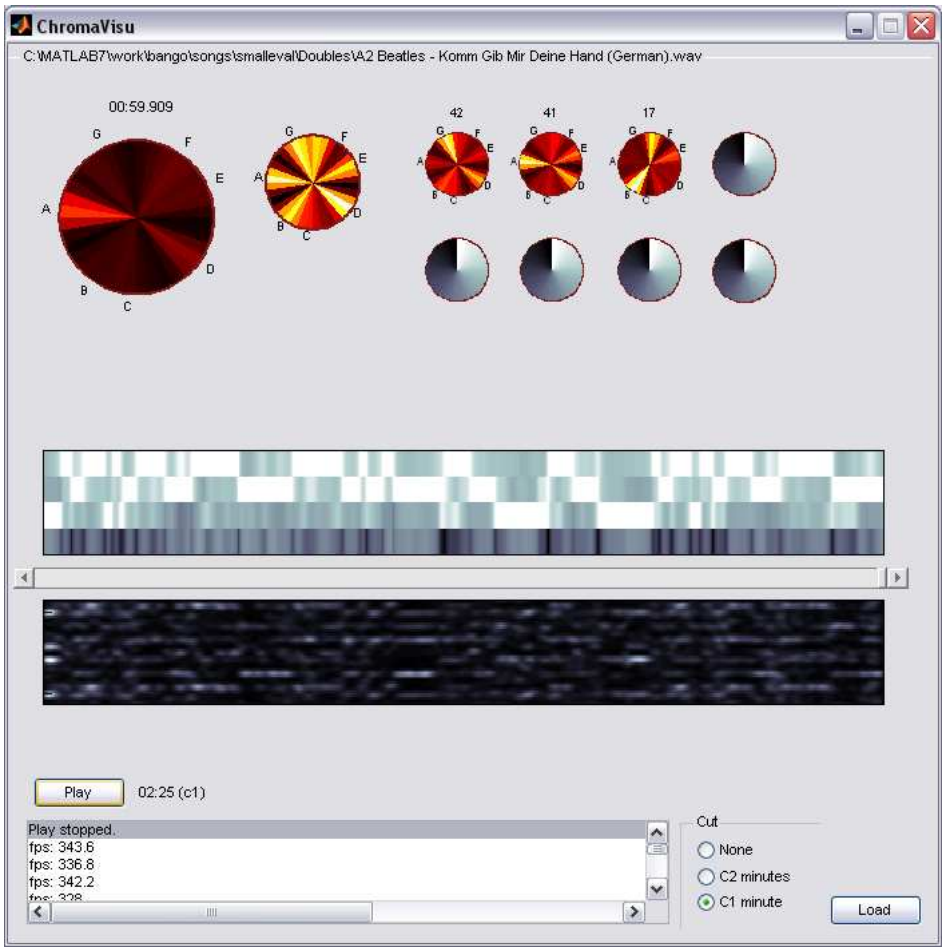


Figure 8.1: ChromaVisu: a tool to study chroma pattern complexity.

E. Just below the cluster assignment is a slider which helps track the current position of the song.

F. Below it is the chromagram in a flat representation. The first five rows and the last five rows are repetitions to help recognize patterns on the boundaries.

8.5.4 ChromaVisu Results

In this section we illustrate the chroma complexity for several pieces from different styles. The main characteristic is the number of different chroma patterns. In the current implementation this number is limited to the range 2-8. As can be seen for some genres this number is relatively high while others tend to have lower numbers. However, the examples also demonstrate that there are always exceptions, thus chroma complexity by itself is not a suitable similarity measure and needs to be combined with additional information (such as rhythmic patterns) for applications such as playlist generation.

We chose typical examples from the categories: jazz, classic piano, classic orchestra, dance, hip hop, and pop. From each piece we analyze the center minute.

As can be seen the distinction between pieces with low complexity and pieces with high complexity is meaningful. Compared to the MFCCs used for spectral similarity the chromagram is a high-level representation of a song. Using chromagram-based information such as the chroma complexity, we might be able to reach beyond the glass ceiling pointed out in D3.1.1 and D3.2.1.

Despite promising results the work is still very preliminary. A obvious next step is to start from a higher level representation of the chromagram. We describe this after the description of the ChromaVisu results.

Jazz (Dave Brubeck Quartet)

The chroma complexity usually ranges from 7-8. However, there are exceptions such as "Take Five" shown in Figure 8.3. In the analyzed section from this piece the drummer plays interesting and complex patterns while the other instruments (including the piano) are in a loop. This illustrates the need for additional descriptors. Alternatively, it would also make sense to analyze the whole piece instead of only one minute.

In general the patterns in the cluster assignment are rather complex. This suggests that a descriptor based on the texture might be a useful supplement to the number of patterns.

Classic Orchestra

The chroma complexity usually ranges from 6-7. The values are generally high, but not as high as for jazz. This is also reflected in the complexity of the cluster assignment which seems to be structured clearer.

Classic Piano

The chroma complexity usually ranges from 7-8. The values are comparable to those of jazz. Some of the patterns in the cluster assignment are as complex as those of the jazz pieces, others are similar to those of classical orchestra.

Dance

The chroma complexity usually ranges from 2-4. The patterns in the cluster assignment are often very simple repetitions. A frequent observation is that the same chroma pattern represents a larger part of the piece as can be seen in Figure 8.6.

Hip Hop

The chroma complexity for hip hop is usually around 2. The variations in this style of music are based on the lyrics, rhythm, beats, melody, but seldomly on the harmonic structure. In Figure 8.7 the two chroma patterns found are very similar. The main difference is that in the second pattern (which occurs 29% of the time) D is stronger pronounced. If the system were not limited to find at least 2 patterns it is possible that the two would be summarized by one.

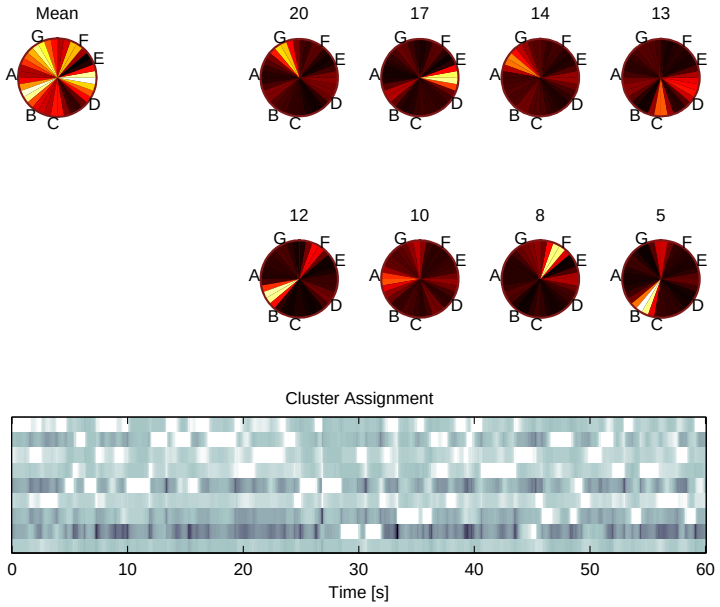


Figure 8.2: Jazz, Dave Brubeck Quartet, Strange Meadow Lark.

Pop

The chroma complexity for pop is usually around 2-4. The patterns in the cluster assignment are often simple repetitions. For example, in Figure 8.8 the cluster assignment reveals that the sequence 1,2,3 is repeated frequently.

8.5.5 ChromaVisu2 Tool

A step towards higher level analysis of the chroma patterns and their relationship to each other is to use a Notennetz representation. While the representation used in the previous visualization is basically one-dimensional the Notennetz is (at least) two-dimensional. The notes are arranged according to major and minor thirds as well as fifths. Based on this representation it is, e.g., easy to identify major and minor chords as rectangles pointing upwards or downwards respectively. The basic layout of a Tonnetz is shown in Figure 8.9.

To study opportunities based on this representation we implemented a visualization tool which allows us to listen to the music while the patterns change (a screenshot is shown in Figure 8.10).

A major difficulty is that the maximum distance between two notes on the Tonnetz is 2 (e.g. between C and C#). To address this it appears to be useful to consider the original four-dimensional space from which the Tonnetz is derived. This is discussed in the next section. Another problem is how to represent the complex patterns (e.g. triangles) for machine learning approaches. Again, this will be discussed in the next section.

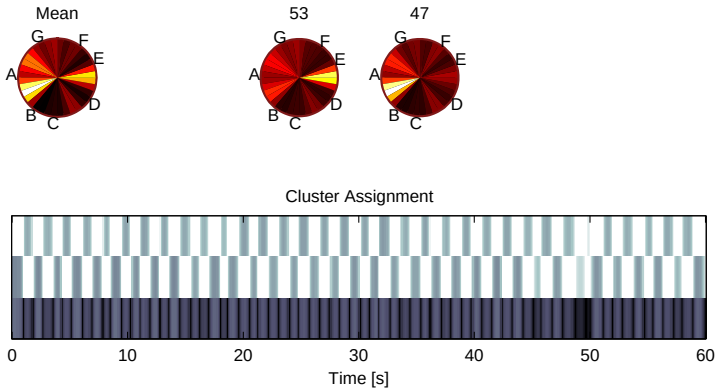


Figure 8.3: Jazz, Dave Brubeck Quartet, Take Five.

8.5.6 ChromaVisu3 Tool

One option to represent a triangle with a single value is its center of gravity. However, a perhaps more meaning ful way to represent the data than the Tonnetz exists. In particular, the notes can be described in terms of the circle of fifths (i.e. two dimensions) and in terms of either a circle of major or minor thirds (i.e. another two dimensions).

Computing the center of gravity in this four dimensional space is a more accurate representation of chords as it allows a higher number of variations. However, the four-dimensional space is difficult to visualize. A three dimensional projection on a torus is possible, however a curve with complex patterns in the three dimensions is difficult to visualize.

A simple solution is to use a linear projection onto two-dimensions which preserves as much of the variance of the four-dimensional data as possible (this is also known as principal component analysis). This curve can then easily be studied and used for further exploratory investigations of the data and possible high-level descriptors.

Figure 8.11 shows the tool we implemented to study the curves in the four-dimensional space. The overall curve is visible as gray line. The current position is marked by red dots. In this example, the two-dimensional projection preserves about 80% of the variance. (Note that the x and y-axis are a linear combination of the four dimensions and cannot easily be interpreted directly.) The curve first moves between the upper left and the lower center, and then in the second part of the piece between the lower center and the upper right. The red dots remain in an area if a chord is stable. Based on the structure of the curve is appears possible to distinguish between more complex and simpler pieces. Future work will aim at ensuring that the curve is not too sensitive to the melody of the piece and studying the curve pattern over longer time frames.

8.6 Conclusions and Future Work

In this section we have demonstrated an approach to use mid-level descriptors developed in WP2 for WP3 tasks. Specifically, we have used the chromagram to develop a measure for chroma complexity (which is closely related to chord

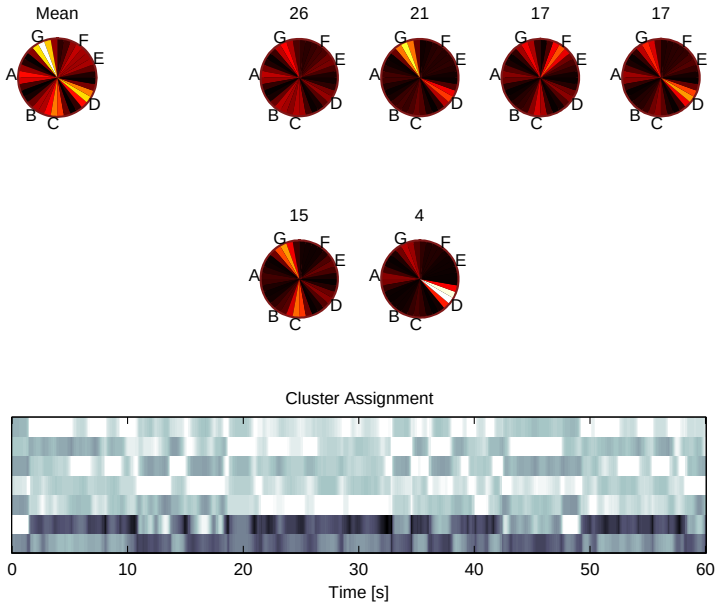


Figure 8.4: Classic Orchestra, Ravel Maurice, Bolero.

complexity). Although the results are preliminary, they indicate that the glass ceiling pointed out in D3.1.1 and D.3.2.1 might be higher than originally anticipated.

As shown with the Take Five example it is necessary to combine chroma complexity with additional (complementary) similarity measures including spectral similarity. This combination could be done as demonstrated in D3.4.1.

Including as much musical knowledge as possible improves the representations and will benefit music similarity measures. Future work will focus on how higher level descriptors can be used for genre classifcation and playlist generation.

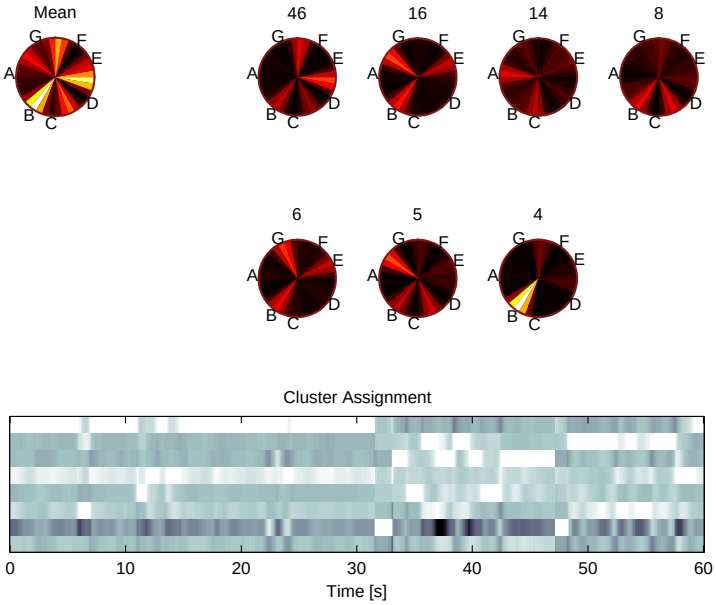


Figure 8.5: Classic Piano, Chopin, Etude, Op 25, No 7.

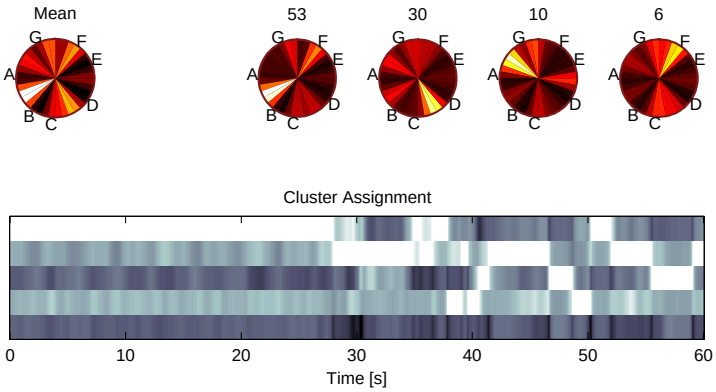


Figure 8.6: Dance, DJs at Work, Time to Wonder.

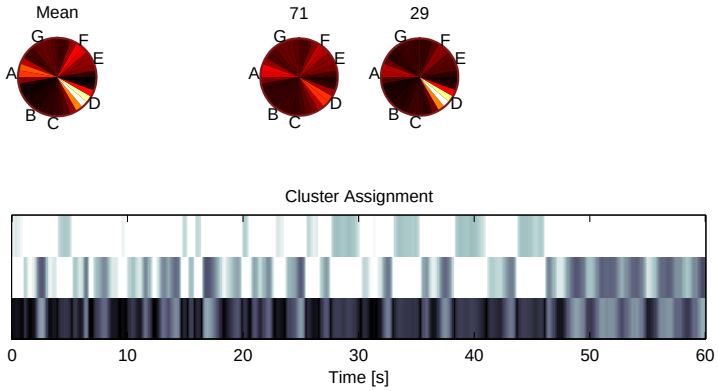


Figure 8.7: Hip Hop, Nelly, EII.

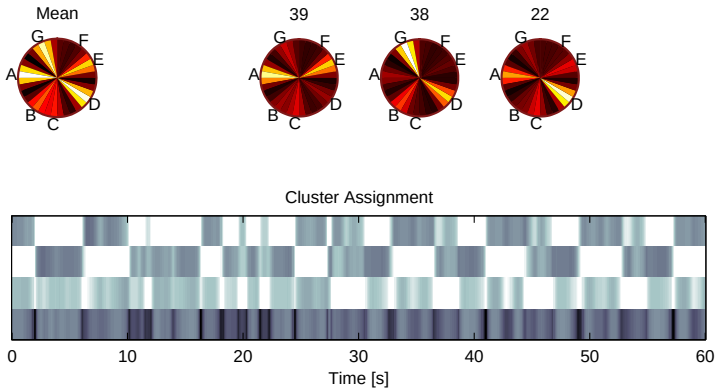


Figure 8.8: Pop, Emma Bunton, What took you so long?

Figure 8.9: Tonnetz.

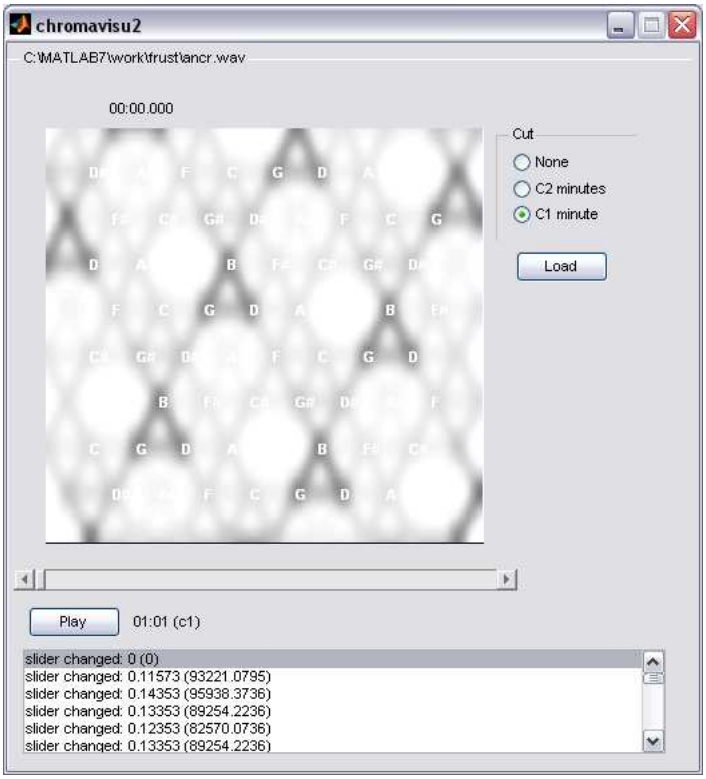


Figure 8.10: Screenshot of ChromaVisu2. A G-major chord is visible.

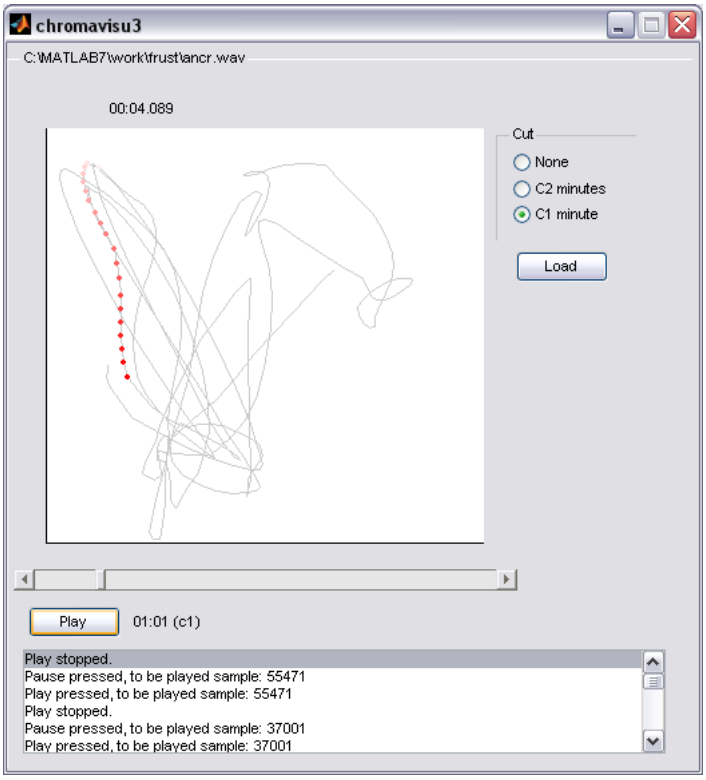


Figure 8.11: Screenshot of ChromaVisu3.

Appendix A

An open source tool for semi-automatic rhythmic annotation

Rhythm is a fundamental musical feature. Anyone perceives rhythm while enjoying music listening. One can represent rhythm explicitly (i.e. write it down) in many ways, with diverse degrees of detail [GM03] and by different means, manually or automatically. For instance, a trained listener can transcribe a musical piece into score notation while listening repeatedly to it. He can also assign a single value for the basic tempo (in BPM). The level of detail in the representation depends on the purpose of annotation. That is, different applications require different representations [GM03].

In any case, it is clear that the task of associating such metadata to musical pieces would be eased by the use of additional software tools. For instance, a simple sound editor plotting waveform and spectrogram would be highly informative to a potential user. Also, a system that would compute automatically the desired metadata would obviously be relevant. However, in this case, subsequent human corrections are a must. Further, as it is clear that no automatic rhythm description system is perfect, nor human annotations are error-free, interactive systems are highly desirable. In such systems, either the user or an algorithm does a first rough analysis of (part of) the data, then the other uses the results of this analysis to orient its own analysis; the process can be iterated several times.

Very few beat annotation systems exist. In [GM97b], Goto refers to a "beat-position editor." This is a manual beat annotation tool that provides waveform visualisation and, for accurate annotations, audio feedback in the form of short bursts of noise added at beat times. To our knowledge, the only publically available (and open-source) beat annotation software is BeatRoot [Dix01d]. To lower the annotation effort, an automatic beat tracking algorithm is available. Interactivity resides in that the user's corrections to the algorithm output (the beat times) are fed back as inputs to the very algorithm.

In this work, we report on a system built upon BeatRoot as well as other open source audio processing tools, namely WaveSurfer [SB00] and CLAM (both part of the AGNULA GPL distribution of Linux sound software). The intent is to "take the best of several worlds", that is, group useful functionalities of those different softwares in a single application as well as expand their scope and capabilities.

We focus on a particular kind of rhythmic annotations, the metrical structure, as it has been formalised by Lerdahl and Jackendoff in the Generative Theory of Tonal Music [LJ83b]. That is, the metadata we propose to associate to musical signals are particular time points: the beats, at several metrical levels.

Annotations can be stored locally and, when correct, they can easily be uploaded to a distant repository, e.g. a structured musical metadata database such as the MTG database [CKF+04b], via the SOAP protocol.

A.0.1 Applications

The knowledge of beats at different levels of the metrical hierarchy can be useful in many applications.

In Music Information Retrieval research, metadata associated to musical data are very useful. First of all because a database of "ground truth" metadata greatly facilitates the design of automatic algorithms for audio content description. In addition, some recent work in this field includes rhythmic infor-

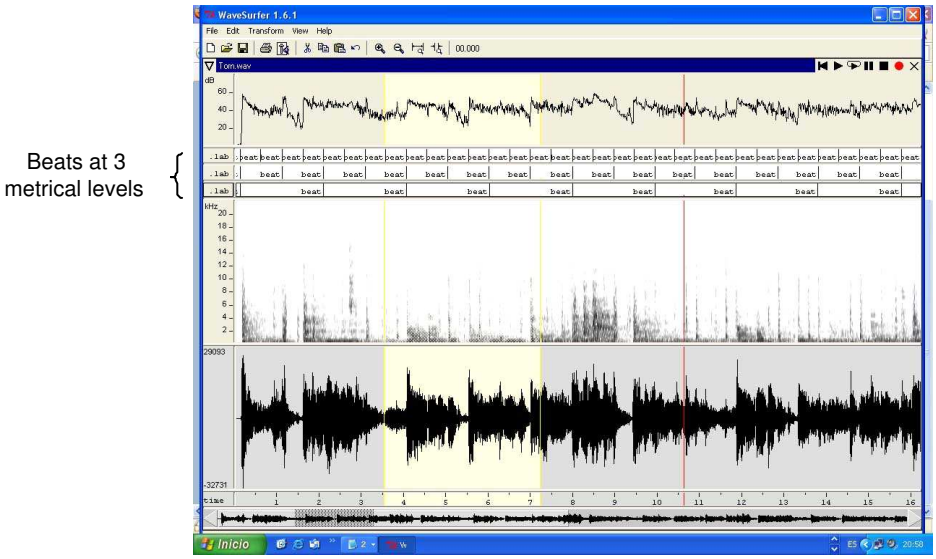


Figure A.1: *Screenshot of metrical level annotations associated to common WaveSurfer functionalities (power plot, spectrogram, waveform, timeline, etc.). This annotation took less than one minute.*

mation as input to systems that compute other types of metadata. For instance, beats at a metrical level can be used to determine other metrical levels [GM99b], [GH03], [Kla03c]. They can also be useful as audio segment boundaries for instrument classification, such as percussion [GH01b], [PK03d], [UD04]. Other examples are the use of the metrical structure for long-term segmentations and rhythmic complexity computation. However, reliable determination of such information from automatic systems is itself a challenge. It is therefore clear that in this type of research, semi-automatic systems would be desirable.

Performers' choices in tempo and expressive timing with respect to position in the metrical structure are very relevant to Musical Performance research. Software tools that ease the annotation of the whole metrical structure, and that generate tempo or timing deviation curves are clearly useful in this field [DGW02].

Finally, other applications are the synchronisation or the sequencing of several musical excerpts, the determination of "looping points" for cut-and-paste operations, the application of tempo-synchronous audio effects (or visual animations), music identification, rhythmic expressiveness transformations.

A.0.2 Annotating metrical levels semi-automatically

This section details in diverse use cases the diverse functionalities offered by the system. Some functionalities are available in WaveSurfer (common sound editing) and others have been added (beat tracking and database connections). Figure A.1 gives an illustration of one possible configuration for the system and the result of the annotation of three metrical levels.

WaveSurfer functionalities

WaveSurfer [SB00] was initially developed as an open source software for speech research at the Department of Speech, Music and Hearing at the Royal Institute of Technology in Sweden.[1] We found diverse reasons to build a rhythmic annotation system on top of it.

First of all, WaveSurfer's typical applications are sound analysis and annotation/transcription. It therefore offers many useful functionalities such as visualisation of waveform, spectrogram, power plots, pitch contour, formant plots, etc. Panes (for transcription, data visualisation, etc.) can be dynamically added or removed, they are all time-aligned and display a running cursor while playing. Collection of panes can be saved as a configuration, that can be applied later to any other sound. This allows users to easily customize the interface. The complete sound waveform is displayed at the bottom while (optionally) a separate pane displays solely part of the waveform with additional zooming functionalities. This greatly simplifies working with large sound files. Data plots are also available, opening the way to easily visualise any relevant data, e.g. tempo curves, deviation curves. WaveSurfer has been designed in agreement with common user interface standards and it also provides intuitive keyboard shortcuts for playing, stopping, selecting regions, looping them etc. It is also possible to easily customise the look-and-feel to personal tastes.

Then, importantly, this is a multi-platform software and it handles many standard audio file formats. Last, but obviously not least, it is extensible through a simple plug-in architecture (note that it is widely used and diverse functionalities are regularly added by researchers in the audio processing community).

Diverse annotation modes

Manual annotations The user can set manually the time indexes of every beat by simple left-clicks on the computer mouse. However this task is very time-consuming and error-prone. Therefore we added the possibility to specify beat times in real time while listening to the sound by simply tapping any of the keyboard's key.

Individual beats can be subsequently adjusted with the help of the computer mouse by selecting and finely shifting them.

Interactive annotations The system presents several menus in addition to WaveSurfer's usual ones as found in transcription panes:

- 'Compute beats'
- 'Erase beats from cursor'
- 'Retrack beats'
- 'Play clicks on beats'
- 'Download tempo and compute beats'
- 'Upload beats to MTGDB'

[1] http://www.speech.kth.se/WaveSurfer/

- 'Download beats from MTGDB'

Those menus enable interactions between the user and the beat-tracking algorithm. The user can run the algorithm "from scratch" and check its output (i.e. finely adjust the position of each beat). Another option is to "give an orientation" to the automatic computation of beats by providing few, very reliable, information, typically a couple of beats. The user can specify those few initial beats manually, or he can select them among the output of a first run of the algorithm. This is very useful to correct phase errors[2] and confusions between metrical levels.

The possibility to keep beats up to a certain point and discard the following ones is especially useful in the case of excerpts with strong tempo changes.

In case the excerpt at hand has been downloaded from the MTG database, it might have a tempo value (in BPM) associated to it (some assets in the current repository are already characterised by an approximate value for the tempo). This metadata is clearly a very useful input to the algorithm.

As in [GM97b] and [Dix01d], short audio signals can be added at beat times for useful audio feedback, for instance noise bursts, closed hi-hat or cowbell samples.

Annotating several metrical levels

Several metrical levels coexist in music, from low levels (small time divisions) to high levels (longer time divisions). And a beat at a high level must also be a beat at each lower level [LJ83b].

Therefore, provided that a metrical level has already been annotated, the process of specifying the beats of the next higher metrical level is simple: create a new transcription pane and copy already annotated beats (i.e. the lower level), select a few beats that determine the next level and discard the others.

When determining a low level from a higher one, the same process applies, but instead of discarding beats, the user can specify whether a duple or a triple subdivision should be performed. Alternatively, it is possible to add a few beats in between successive beats, the data can then be retracked with this newly defined metrical level.

Differences with BeatRoot

The graphical interface shows important differences with BeatRoot. For instance, it is our belief that WaveSurfer's built-in visualisation functionalities (e.g. running cursor and scrolling panes synchronized with audio playback), and its intuitive keyboard shortcuts and general look-and-feel are an enhancement of BeatRoot's interface. Also, an important point is that graphical configurations can be defined by the user. Other relevant differentiating features are the capture of keyboard messages while listening to the audio, as well as database connection facilities, multi-platform support[3] and the possibility to instantiate diverse transcription panes in order to annotate several metrical levels. However, it must be noted that BeatRoot also permits to annotate beats of MIDI data, which the system reported here does not permit.

[2] i.e. when the tempo is correctly computed but the computed beats are all a constant away from their correct positions

[3] i.e. Debian Linux and Windows XP at proceedings publishing date

A.0.3 Implementation

General architecture

WaveSurfer is built using Snack, a sound manipulation extension to the Tcl/Tk scripting language. Snack can be used as a scripting language, from a command prompt, WaveSurfer offers a graphical interface to it. See [SB00] for details. The important point here is that functionalities can be added to WaveSurfer by creating new Snack (or Tcl) commands and calling them through plugins.

In order to add beat tracking functionalities to WaveSurfer, two components are needed: an *external library* embedding those functionalities (with an appropriate C interface)[4] and a *plugin script* to call them from WaveSurfer.

The plugin script is written in Snack and logically accounts for a command that loads the library, typically:
'load C:/.WaveSurfer/1.6/plugins/libBeatSnack.dll' (in Windows).
Additional commands (calling library functions) are directly accessible from WaveSurfer transcription panes, as e.g. 'Compute beats', 'Retrack', 'Erase beats from cursor', etc.

On the other hand, the external library is written in C and C++ (see Figure A.2). A first part of the library creates the actual Snack function called in the plugin e.g. from the command 'Compute beats'. This part is written in C. The second part, the core beat tracking algorithm, is written in standard C++ and CLAM.

As shown in Figure A.2, Interface2CLAM.h is the interface between these two parts of the library. This interface is a simple function whose input parameters are the sound samples and optionally a list of beats. Its output is a beat list (and optionally an onset list). The function definition is written in Interface2CLAM.h while its implementation is in Interface2BeatRoot.cxx.

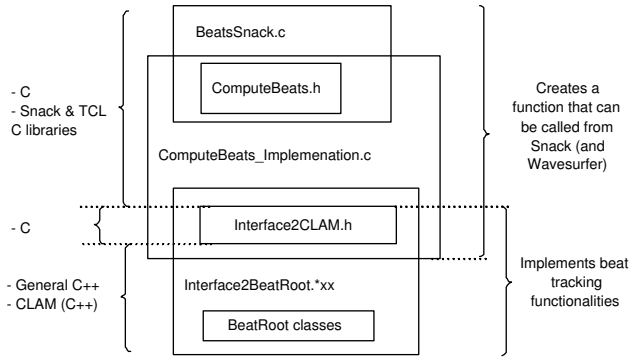


Figure A.2: *External library architecture.*

Beat tracking algorithm

The automatic algorithm at the core of the system is the association of the transient detection and tempo induction algorithms already present in the CLAM

[4]see http://www.speech.kth.se/snack/modex.html

library and BeatRoot's beat tracking method [Dix01d].[5] This part of BeatRoot has been ported to the CLAM framework, and in the current version, the tracking part is intended not to present differences with the public version of BeatRoot (the set of default parameter may differ though).

In order to account for the diverse usage modes specified above, the input to the beat tracking algorithm can be the following:

- Case 1: The sound samples.
- Case 2: The samples and some (correct) beat times.
- Case 3: The samples and a (correct) tempo value

In all cases, outputs are beat times of the whole audio.

Database connection

In case the audio signal under annotation is part of the MTG repository (see [CKF+04b]), interesting functionalities are provided.

The audio repository can be browsed from any remote web browser. It is possible to launch WaveSurfer directly from the web interface. The audio is loaded automatically together with annotations that may be available on the repository. There is a connection to the MTG database server via web services (SOAP) which allows to upload new segmentations. SOAP[6] is a lightweight protocol based on XML-RPC calls. The Service interface is described in a WSDL[7] file (a superset of XML) indicating methods calls, objects, and exceptions that will be sent across the net. As all the exchange of information is made with XML (both data and control messages), SOAP allows the interaction between programs on different platforms or in different languages running anywhere on the Internet.

A.0.4 Summary

We presented a multi-platform software for semi-automatic beat annotation of audio signals. Being open source, it can be used and modified at will. It has been implemented as a plugin for the WaveSurfer sound editor and, in addition to this software's functionalities, it embeds useful functionalities from CLAM and BeatRoot. Source code and binaries are available from:
http://www.iua.upf.es/~fgouyon/BeatTrackingPlugin.html

At the time of writing, additional functionalities are being added to this software. For instance the handling of diverse audio file format (WaveSurfer can handle many standard format (WAV, AIFF, MP3, etc.), so does CLAM, however, we still did not add this functionality to the beat tracking algorithm). We are also working on the integration of the free "aubio" library developed by Paul Brossier at Queen Mary University of London,[8] this library provides a C implementation of powerful and fast onset detection algorithms [BDD+04]. Visualisation and interactive onset detection will also be a useful feature. Finally, we are working on the addition of diverse data plots such as tempo curves.

[5] http://www.oefai.at/~simon/beatroot/index.html
[6] http://www.w3.org/TR/soap/
[7] http://www.w3.org/TR/wsdl
[8] http://piem.homeip.net/~piem/aubio

Appendix B

Chord annotation: a case study

There is currently no standard methodology for chord annotation. When designing chord detection algorithms, the lack of annotated databases makes evaluation and comparison of results difficult. We attempt to address this problem by defining a rigidly structured general annotation system for chords. Such an annotation system will afford researchers the opportunity to share annotated files easily. The system presented here is easy for musically trained individuals to write and understand, yet simple and unambiguous to parse with computer programs.

In classical western harmony notation, a style developed for score analysis, certain chord symbols depend upon the musical key context for their full meaning to be apparent. In contrast, jazz and popular music notations are more commonly used for performance and are generally more explicit in their meaning to avoid being misread. Text file annotations are often a straight translation of an individual's preferred musical notation to the nearest textual equivalent. This lack of standardisation can cause many problems when other people come to read and interpret the annotated symbols. To address this problem, we introduce a general logical model for a musical chord in Section B.0.6. This is used in Section B.0.7 to define the rules and syntax for a representation for chords in flat text with a formalised description of the syntax given in Backus-Naur Form (BNF) [LM81].

B.0.5 Properties of Musical Chords

When two or more notes are played simultaneously, a chord is produced. In Western tonal music, any musical chord may be represented with the following information:

- The *root* note of the chord; the note upon which the chord is built.
- Its type or *quality*, defined by the component intervals that make up the chord relative to the root.
- Its *inversion*, defined by the degree of the chord played as its bass note.

These parameters remain consistent for all the different ways in which notes of a particular chord may be played, or *voiced* [Tay89].

Styles of Notation

There are many styles of harmony and chord notation in music. These conventions can vary not only across genres but also within them. To illustrate some of the variation in chord notation methods, several styles are shown for the short excerpt of music in Figure B.1(a).

Figured Bass

The first style, in Figure B.1(b) is the Baroque *Figured Bass*. This was a system of figures written underneath a bass line indicating which intervals should be played above the bass note to complete the correct harmony [Tay89].

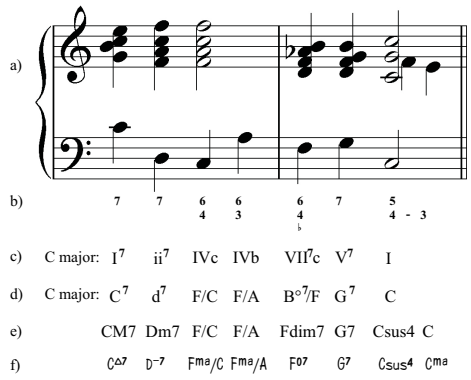


Figure B.1: A short extract of music in C major with different harmony notations: a) Musical score b) Figured bass, c) Classical Roman numeral, d) Classical letter, e) Typical Popular music guitar style, f) Typical jazz notation

Classical Harmony Analysis

In classical Western harmony analysis, chord notation was developed to show the sequential aspects of harmony or *harmonic progression* rather than just the particular chord or sonority at any given instant [Tag03]. Figure B.1(c) shows Roman numeral style notation. Chords are labelled according to the position of their root note within the scale related to the current key [Tay89]. Inversions are marked with 'b' for first inversion, 'c' for second inversion and so on if the chord has further degrees. The notation shown in Figure B.1(d) with letters denoting the root notes of chords is also common in classical analysis. In both cases major chords are shown with uppercase characters and minor chords in lowercase.

In classical notation, because chords are notated in the context of a given key, certain properties are implied rather than explicitly marked. For example, in a major key, the seventh degree of the key scale is a major seventh interval, so in marking a tonic major seventh chord 'I^7' with a superscript 7, the major seventh is implied. However, a dominant seventh chord, by definition, contains a minor seventh interval but it is also marked with a superscript 7 'V^7' (see second bar of the example in Figure B.1(c)). In the Roman numerals system it is clear that 'V^7' is a dominant chord but when using letters as shown in Figure B.1(d) this can become a source of ambiguity. The extract is in the key of C major so the first chord is marked C^7 but the dominant chord in the second bar is marked G^7. If the key context is lost from this notation, which is a possibility if storing these symbols in a text file, then there can be no sure way of telling which quality of seventh chord the transcriber intended without trying to infer the context from the chord progression.

Jazz and Popular Music

In popular music and jazz, the role of chord symbols is more tailored for use in performance, with jazz musicians in particular often playing at sight. For this reason chords are notated in a much more explicit manner so that musicians

need spend the minimum of time and thought to correctly work out what they are required to play. The qualities of chords are marked explicitly but the markings that are used vary widely and it is hard to find two people who agree on a preferred style for every chord type.

The first chord of the example in Figure B.1(a), a C major seventh, may be marked as CM7, CMaj7, or $C^{\triangle 7}$ [Cok64] as seen in Figure B.1(d) and B.1(e). The second chord in the example, a D minor seventh, may be marked Dm7, Dmin7 and D^{-7}. The G seventh chord in the second bar can be marked G7 or G^7 or sometimes Gdom7, although this last marking is often incorrectly applied in cases where the seventh chord does not actually function as a dominant chord. Inversions are most often denoted by an oblique stroke (/) followed by the bass note to be played. This can be seen with the inverted F major chords, F/C and F/A, at the end of the first bar of the example [Tay89].

Ambiguity between chord symbols can occur when translated to flat text if the notation convention used by the transcriber is not given. For example, if an annotation contains the symbol `A7`, this could be a seventh chord in jazz notation or in classical notation if in the key of D. However, it could also be a major seventh chord in classical notation if in the keys of A or E major. It is to avoid this kind of ambiguity that we propose the adoption of the chord symbol representation outlined in the following sections.

B.0.6 A Model for musical chords

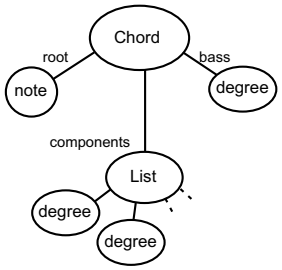


Figure B.2: Model for chord definition

We define a model to represent chords unambiguously and independent of key context. The root is defined as a *note* element which has an absolute pitch class value. The list of component intervals and the bass note are defined as *degrees*, relative to the root note. A diagram of this model is shown in Figure B.2.

We define seven *natural* note names (letters A to G, eqn. B.1), which correspond to the white keys on a piano keyboard. We also define thirteen *intervals* (numbers 1 to 13, eqn. B.2), which correspond to the major diatonic intervals (i.e. they are either major or perfect) up to one octave plus a sixth (shown in Figure B.3).

To allow correct spelling of enharmonics we also define two *modifier* opera-

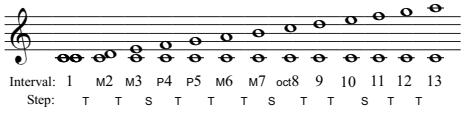


Figure B.3: The Major diatonic intervals upon middle C. 'T' denotes a step of a tone between adjacent intervals and 'S' a semitone.

tors, sharp and flat. Thus:

$$\begin{aligned} \textit{natural} &= \{\text{A} \mid \text{B} \mid \text{C} \mid \text{D} \mid \text{E} \mid \text{F} \mid \text{G}\} && \text{(B.1)} \\ \textit{interval} &= \{1 \mid 2 \mid 3 \cdots 11 \mid 12 \mid 13\} && \text{(B.2)} \\ \textit{modifier} &= \text{sharp} \mid \text{flat} && \text{(B.3)} \end{aligned}$$

Naturals and intervals may be operated on by these modifiers. In this way, notes and degrees may be defined as:

$$\begin{aligned} \textit{note} &= \textit{natural} \mid \textit{modifier}(\textit{note}) && \text{(B.4)} \\ \textit{degree} &= \textit{interval} \mid \textit{modifier}(\textit{degree}) && \text{(B.5)} \end{aligned}$$

An example model of a chord is shown in Figure B.4. The chord in the example is a C minor seventh chord in first inversion. The root of this chord is a C. The component intervals are a minor third, a perfect fifth and a minor seventh (♭3, 5, ♭7). The bass note of a first inversion chord is its 3rd degree, which in this example is an E♭.

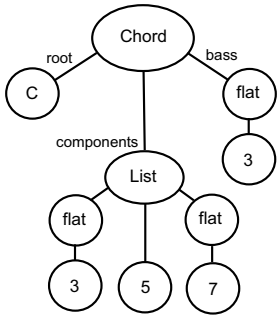


Figure B.4: Example model of a first inversion C minor-seventh chord

The sharp and flat modifiers allow proper enharmonic spelling of notes and intervals. This is important in cases such as the diminished seventh chord (comprising the musical intervals ♭3, ♭5, ♭♭7) which contains a diminished seventh interval (a major seventh interval flattened twice). Although this interval is tonally equivalent to a major sixth, it has a different musical function.

B.0.7 Developing a Syntax for Chord Notation

It is important for use in text annotation that chord symbols be context independent. Using the chord model described in Section B.0.6 and a context

independent approach to notation, similar to the Jazz style [Tay89], we define the following syntax for representing a chord in flat text:

```
root : (degree1, degree2...) / bass
```

The root note is written first followed by a colon (`:`) separator. A comma delimited list of the chord degrees is then written, contained by parentheses. Finally, an optional bass note may be added at the end after a forward slash character (`/`) if it is different to the root. The naturals, intervals and modifiers are defined in Table B.1 following equations B.1 to B.3. The sharp and flat are signified by the hash symbol `#` and the lowercase `b` respectively.

To keep the notation musically intuitive, note modifiers come after naturals so A♭ becomes `Ab`. Degree modifiers come before intervals so a flattened seventh becomes `b7`. An extra chord state denoted by a single uppercase `N` is also added to signify 'no chord' to mark silence or untuned, possibly percussive musical material. To resolve the possible ambiguity between a note `B` and a flat modifier `b` the notation is necessarily case sensitive.

Following these rules, all chords may now be described in flat text in an unambiguous manner. For example, using our system a C major chord becomes:

```
C:(3,5)
```

Likewise, a C minor chord becomes:

```
C:(b3,5)
```

A more complex chord such as a D♯ minor seventh chord in second inversion with an added ninth would become:

```
D#:(b3,5,b7,9)/5
```

B.0.8 Shorthand Notation

Our chord representation is straightforward and capable of fully describing any chord within Western tonal music. However, for manual annotation purposes, the inclusion of more musically intuitive shorthand labels for common chords is a useful extension. A proposed vocabulary of shorthand labels is given in Table B.2 where each label is understood as a pre-set list of degrees. In this way, a chord may now also be defined by:

```
root : shorthand(extra-degrees) / bass
```

A common convention for labelling the quality of chords is: major 'M', minor 'm', augmented '+' and diminished 'o'. We choose more verbose labels, however, because it makes typographic errors in annotations easier to detect. Provision for extra degrees in parentheses is left so that additional intervals may be added to common chords. To make the shorthand system more flexible a special 'omit degree' symbol, an asterisk `*`, is also added to denote a missing interval from a shorthand notated chord. Hence a C minor seventh chord could become:

```
C:min7    ≡    C:(b3,5,b7)
```

and a C minor seventh with an added 11th degree but no 5th degree could be written:

`C:min7(*5,11)` ≡ `C:(b3,b7,11)`

To stay consistent with most chord notation styles, a root note on its own (i.e. with no shorthand label or defined degrees) is assumed to denote a major chord. Therefore a C major chord may be written simply as:

`C` ≡ `C:maj` ≡ `C:(3,5)`

Likewise, a root note followed directly by a forward slash and a bass note is assumed to be a major chord in an inverted form. For example a first inversion A major chord could be written:

`A/3` ≡ `A:maj/3` ≡ `A:(3,5)/3`

Added note chords should be explicitly labelled as major or minor to avoid confusion. Therefore a C major with an added fourth becomes:

`C:maj(4)` ≡ `C:(3,4,5)`

Table B.1: Syntax of Chord Notation in Backus-Naur Form

```
<chord>        ::=  <note> ":" [<shorthand>]["("<degree-list>")"]["/"<degree>]
                  | <note> ["/"<degree>]  | "N"

<note>         ::= <natural> | <note> <modifier>

<natural>      ::= A | B | C | D | E | F | G

<modifier>     ::= b | #

<degree-list>  ::= ["*"] <degree> ["," <degree-list>]

<degree>       ::= <interval> | <modifier> <degree>

<interval>     ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13

<shorthand>    ::= maj | min | dim | aug | maj7 | min7 | 7 | dim7 | hdim7
                  | minmaj7 | maj6 | min6 | 9 | maj9 | min9 | sus4
```

Table B.2: Shorthand definitions for common chords

Chord Type		Shorthand Notation	Components List
Triad Chords:			
	Major	`maj`	`(3,5)`
	Minor	`min`	`(b3,5)`
	Diminished	`dim`	`(b3,b5)`
	Augmented	`aug`	`(3,#5)`
Seventh Chords:			
	Major Seventh	`maj7`	`(3,5,7)`
	Minor Seventh	`min7`	`(b3,5,b7)`
	Seventh	`7`	`(3,5,b7)`
	Diminished Seventh	`dim7`	`(b3,b5,bb7)`
	Half Diminished Seventh	`hdim7`	`(b3,b5,b7)`
	Minor (Major Seventh)	`minmaj7`	`(b3,5,7)`
Sixth Chords:			
	Major Sixth	`maj6`	`(3,5,6)`
	Minor Sixth	`min6`	`(b3,5,6)`
Extended Chords:			
	Ninth	`9`	`(3,5,b7,9)`
	Major Ninth	`maj9`	`(3,5,7,9)`
	Minor Ninth	`min9`	`(b3,5,b7,9)`
Suspended Chords:			
	Suspended 4th	`sus4`	`(4,5)`

Appendix C

Annotation via Automatic Alignment of Audio

An algorithm was developed for aligning audio recordings of different performances of the same piece of music, based on an efficient implementation of a dynamic time warping (DTW) algorithm. This is useful for precise content-based indexing of audio files where one rendition of each piece is already annotated, so that after the alignment is completed, the metadata (e.g. section, phrase, beat or note indexes) can be transferred automatically from one recording to the corresponding positions in the other recording. The audio alignment algorithm is also useful for performance research, intelligent audio editing software and live visualisation of expressive performances.

A forward path estimation algorithm constrains the alignment path so that dynamic time warping can be performed with time and space costs that are linear in the lengths of the audio files. Frames of audio are represented by a positive spectral difference vector, which emphasises note onsets in the alignment process. The alignment system was tested on several hundred test cases and had a median error of 20 ms, with less than 1% of test cases failing to align at all.

C.0.9 On-Line Time Warping

The quadratic time and space cost is often cited as a limiting factor for the use of DTW with long sequences. However by using path constraints, a linear time and space algorithm can be developed. In constraining the path, the danger is that the optimal path will be excluded. This danger is reduced by using a forward path estimation algorithm to calculate the centre of the band of the cost matrix which is to be calculated. This is based on the on-line time warping algorithm presented in [Dix05b, Dix05a], which estimates the alignment of a live performance with a recording in real time. The DTW path is constrained to lie within a fixed distance of the forward path, which ensures that the computation is bounded by linear time and space costs.

The intuition behind the forward path algorithm can be explained with reference to Figure C.1, where a band width of $w = 4$ is used for illustrative purposes. (In practice, a band width of $w = 500$ is used.) At any time the *active area* of the matrix is the top row and the right column of the calculated area. The minimum cost path to each of these cells is evaluated and the cell with the lowest minimum cost path is used as an indication of the direction in which the optimal path appears to be heading. (The true optimal path cannot be known unless the complete matrix is calculated.) If this cell is in the top right corner, the algorithm is considered to be on target. If it is to the left of the target (for example, after expansions 7 and 8 in Figure C.1), then the calculated part of the matrix is expanded upwards until the algorithm is on target again (expansions 9 to 11). Likewise if the cell is below the target, expansion is performed to the right. A complete description of the forward path algorithm can be found in [Dix05a]. When the ends of both files are reached, the optimal path is traced backwards using the standard DTW algorithm, constrained by the fact that only the cells calculated previously during the forward path calculation can be used.

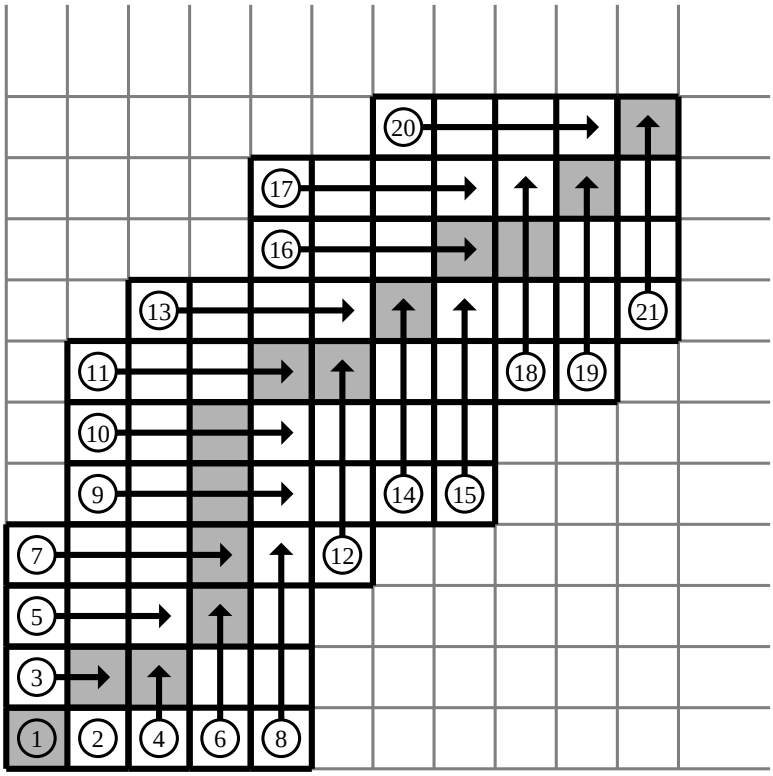


Figure C.1: *An example of the on-line time warping algorithm with band width $w = 4$, showing the order of evaluation for a particular sequence of row and column calculations. The axes represent time in the two files. All calculated cells are framed in bold, and the optimal path is coloured grey.*

C.0.10 Comparison of Audio Frames

The most important factor for alignment is the timing of the onsets of tones. The subsequent evolution of the tone gives little information about the timing and is difficult to align using energy features, which change relatively slowly over time within a note. Frames of audio input are converted to a frequency domain representation using a short time Fourier transform, and then mapped to a non-linear frequency scale (linear at low frequencies and logarithmic at high frequencies). The time derivative (first-order difference) of this spectrum is then half-wave rectified and the resulting vector is employed in the DTW algorithm's match cost function, using a Euclidean metric. The use of first order difference ensures that only the increases in energy in each frequency bin are taken into account, which generally indicate the onsets of tones.

C.0.11 Testing and Results

We report the results from a precise quantitative evaluation using data recorded on a Bösendorfer computer-monitored piano, for which we had both the audio recordings and measurements of the timing (resolution 1.25 ms) and dynamics of every note. We used a set of recordings of 22 pianists playing 2 excerpts of solo piano music by Chopin (Etude in E Major, Op.10, no.3, bars 1–21; and Ballade Op.38, bars 1–45) [Goe01]. The Etude performances ranged from 70.1 to 94.4 seconds duration, and the Ballade ranged from 112.2 to 151.5 seconds, so the differences in execution speeds were by no means trivial. Alignment was performed on all pairs of performances of each piece (a total of $\frac{22\times21}{2} * 2 = 462$ test cases). The error was calculated as the Manhattan distance of the onset times of corresponding notes in both performances from the nearest point on the optimal path calculated by the time warping algorithm. The total error of an alignment path is the average of the point-wise errors over all score events.

Table C.1 shows the distribution of point-wise errors less than or equal to 0, 1, 2, 3, 5, 10, 25 and 50 frames, where the frame spacing was 20 ms. The average and worst case errors are also shown. These results indicate that the system is quite successful, but this was partly due to the fact that the audio recordings were all made under identical conditions (same piano, microphone, room and settings).

Further tests were made with professional piano recordings over the last 50 years, where the beat had been annotated using the interactive beat tracking system BeatRoot [Dix01b, Dix01c]. The median error for each of these pieces was 20 ms (1 frame), except for one piece which had a median error of 40 ms. In a small number of cases (5 out of 175) alignment fails. The spread of errors indicates a low probability of large errors for this data, making the system useful in practice as an annotation tool. Informal tests with guitar, orchestral and popular music suggest that the similarity measure is not restricted to piano tones, but could be generally applicable to different instruments. Further details of these tests are in [DW05].

C.0.12 Conclusion

This section presented an audio alignment algorithm using dynamic time warping based on a low-level representation of the audio data. In cases where the

Error ≤		Cumulative %age	
Frames	Seconds	Etude	Ballade
0	0.00	46.5%	36.6%
1	0.02	84.5%	77.1%
2	0.04	91.1%	88.9%
3	0.06	93.4%	92.5%
5	0.10	96.0%	95.1%
10	0.20	98.8%	97.4%
25	0.50	99.8%	99.1%
50	1.00	100.0%	99.8%
Average Error		23 ms	35 ms
Worst Error		2340 ms	3640 ms

Table C.1: Alignment results shown as cumulative percentages of score events with an error up to the given value.

tracking was successful, it was also very accurate, with a median error of 20 ms, and average errors from 23 ms to 86 ms. For most purposes of the alignment tool, this is easily sufficient. We intend to extend this work to include score-audio alignment, by synthesising a performance from the score and matching it to the audio recording, so that audio files can be automatically annotated with metadata indicating the timing of individual notes.

Bibliography

[ACN+05] Samer Abdallah, Michael Casey, Katy Noland, Christophe Roads, and Mark Sandler. Theory and evaluation of a Bayesian music structure extractor. Submitted to the 6th ISMIR Conference, London, UK, 2005.

[ALP01] G. Agostini, M. Longari, and E. Pollastri. Content-based classification of musical instrument timbres. *International Workshop on Content-Based Multimedia Indexing*, 2001.

[and95] and. An information-maximisation approach to blind separation and blind deconvolution. 7:1129–1156, 1995.

[AP02a] J.-J. Aucouturier and F. Pachet. Music similarity measures: What's the use? In *Proceedings of the 3rd ISMIR, Paris, France.*, pages 157–163, 2002.

[AP02b] Jean-Julien Aucouturier and François Pachet. Music similarity measures: What's the use? In *Proceedings of the Third International Conference on Music Information Retrieval (ISMIR'02)*, pages 157–163, Paris, France, October 2002. IRCAM.

[AS01] J.-J. Aucouturier and M. Sandler. Segmentation of musical signals using hidden markov models. In *Proceedings of the Audio Engineering Society 110th Convention*, May 2001.

[Bag91] L. Baggi. Neurswing: An intelligent workbench for the investigation of swing in jazz. *Computer*, 24(7):60–64, 1991.

[BC94] G. J. Brown and M. Cooke. Perceptual grouping of musical sounds: A computational model. *Journal of New Music Research*, 23(2):107–132, Jun 1994.

[BDD+04] J.P Bello, C. Duxbury, M. E. Davies, , and M. Sandler. On the use of phase and energy for musical onset detection in the complex domain. *IEEE Signal Processing Letters*, 2004.

[BDDS04] J. P. Bello, C. Duxbury, M. E. Davies, and M. B. Sandler. On the use of phase and energy for musical onset detection in the complex domain. *IEEE Signal Processing Letters*, 11(6):553–556, July 2004.

[BE01] A.L. Berenzweig and D.P. Ellis. Locating singing voice segments within musical signals. In *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 119–123, Mohonk, NY, 2001.

[Ber71] Daniel Berlyne. *Aesthetics and psychobiology*. Appleton-Century-Crofts, New York, 1971.

[BGH+95] S. V. Buldyrev, A. L. Goldberger, S. Havlin, R. N. Mantegna, M. E. Matsa, C.-K. Peng, M. Simons, and H. E. Stanley. Long-range correlation properties of coding and noncoding DNA sequences: Genbank analysis. *Physical Review E*, 51(5):5084–5091, 1995.

[BHM01] J. C. Brown, O. Houix, and S. McAdams. *Feature dependance in the automatic identification of musical woodwind instruments*, volume 109. J. Acoust. Soc. Amer., 2001.

[Bil93] J. Bilmes. *Timing is of the Essence: Perceptual and Computational Techniques for Representing, Learning, and Reproducing Expressive Timing in Percussive Rhythm*. Thesis/dissertation, MIT, Cambridge, 1993.

[BLC04] Dan Barry, Bob Lawlor, and Eugene Coyle. Real-time sound source separation: Azimuth discrimination and resynthesis. In *Proceedings of the AES 117th Convention*, San Francisco, CA, USA, 28–31 October 2004.

[BP05] J. P. Bello and J. Pickens. A robust mid-level representation for harmonic content in music signals. In *Proceedings of the Sixth International Conference on Music Information Retrieval, London, UK*, 2005.

[Bre90] Albert S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound*. MIT Press, Cambridge, MA, 1990.

[Bre96] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[Bro91a] J. Brown. Calculation of a constant Q spectral transform. *Journal of the Acoustical Society of America,*, 89(1):425–434, 1991.

[Bro91b] Judith Brown. Calculation of a constant q spectral transform. *Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

[Bro99] J. C. Brown. *Computer identification of musical instruments using pattern recognition with cepstral coefficients as features*, volume 105. J. Acoust. Soc. Amer., 1999.

[BS03a] J.P. Bello and M. Sandler. Phase-based note onset detection for musical signals. In *International Conference on Acoustics, Speech and Signal Processing*, 2003.

[BS03b] Juan P. Bello and Mark Sandler. Phase-based note onset detection for music signals. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP-03*, 2003.

[BW01] M. A. Bartsch and G. H. Wakefield. To catch a chorus: Using chroma-based representations for audio thumbnailing. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. New Paltz, NY,*, 2001.

[BWFW04] B. Burgeth, M. Welk, C. Feddern, and J. Weickert. Morphological operations on matrix-valued images. In *The 8th European Conference on Computer Vision*, pages 115–167, May 2004.

[Car98] Jean-François Cardoso. Blind signal separation: Statistical principles. *Proceedings of the IEEE*, 86(10):2009–2025, October 1998.

[CC98] J.C.C. Chen and A.L.P. Chen. Query by rhythm: An approach for song retrieval in music databases. In *Proceedings of the 8th IEEE International Workshop on Research Issues in Data Engineering*, pages 139–146, 1998.

[CD04] Markus Cremer and Claus Derboven. A system for harmonic ananlysis of polyphonic music. In *Proceedings of the AES 25th International Conference*, pages 115–120, London, UK, 2004.

[CDGW01] E. Cambouropoulos, S. Dixon, W. Goebl, and G. Widmer. Human preferences for tempo smoothness. Proc. International Symposium on Systematic and Comparative Musicology, International Conference on Cognitive Musicology, 2001.

[CDS05a] N. Chétry, Mike Davies, and Mark Sandler. Musical instrument identification using lsf and k-means. *Proc. 118th AES*, 2005.

[CDS05b] Nicolas Chétry, Mike Davies, and Mark Sandler. Musical instrument identification using LSF and k-means. In *Proceedings of the AES 118th Convention*, Barcelona, Spain, 28–31 May 2005.

[CKDH00a] A. T. Cemgil, B. Kappen, P. Desain, and H. Honing. On tempo tracking: Tempogram representation and Kalman filtering. *Journal Of New Music Research*, 29(4), 2000.

[CKDH00b] A.T. Cemgil, B. Kappen, P. Desain, and H. Honing. On tempo tracking: Tempogram representation and Kalman filtering. In *Proceedings of the International Computer Music Conference*, pages 352–355, San Francisco CA, 2000. International Computer Music Association.

[CKF+04a] P. Cano, M. Koppenberger, S. Ferradans, A. Martinez, F. Gouyon, V. Sandvold, V. Tarasov, and N. Wack. Mtg-db: A repository for music audio processing. In *Proceedings of 4th International Conference on Web Delivering of Music*, Barcelona, Spain, 2004.

[CKF+04b] Pedro Cano, Markus Koppenberger, Sira Ferradans, Alvaro Martinez, Fabien Gouyon, Vegard Sandvold, Vadim Tarasov, and Nicolas Wack. MTG-DB: A Test Environment for Music Audio Processing. In *Proc. International Conference on Web Delivering of Music*, 2004.

[CL00] F. Carreras and M. Leman. Automatic harmonic description of musical signals using schema-based chord decomposition. *Journal of New Music Research*, 28(4):310–333, 2000.

[Cla87] E. Clarke. Levels of structure in the organization of musical time. *Contemporary music review*, 2(1):211–238, 1987.

[Cla99] E. Clarke. Rhythm and timing in music. In D. Deutsch, editor, *The Psychology of Music, 2nd edition*, Series in Cognition and Perception. Academic Press, 1999.

[Cok64] Jerry Coker. *Improvising Jazz*. Simon and Schuster, New York, 1964.

[CW82] M. Clynes and J. Walker. Neurobiologic functions of rhythm, time, and pulse in music. In Clynes M. and Walker J., editors, *Music, mind, and brain: The neuropsychology of music*, pages 171–216. Plenum, 1982.

[CW00] M. A. Casey and A. Westner. Separation of mixed audio sources by independent subspace analysis. In *Proc. of the International Computer Music Conference, Berlin, Germany*, 2000.

[Dat] RWC Music Database. Music genre database and musical instrument sound database. *http://staff.aist.go.jp/m.goto/RWC-MDB*.

[DB01] C. Drake and D. Bertrand. The quest for universals in temporal processing of music. *Annals of the New York Academy of Science*, pages 17–27, 2001.

[DBDS03] C. Duxbury, J.P. Bello, M. Davies, and M. Sandler. A combined phase and amplitude based approach to onset detection for audio segmentation. In *Proceedings of the 4th European Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS-03), London, UK.*, 2003.

[DG03] M. Davy and S.J. Godsill. Bayesian harmonic models for musical signal analysis. *Bayesian Statistics*, 7, 2003.

[DGP99] C. Drake, L. Gros, and A. Penel. How fast is that music? the relation between physical and perceived tempo. In Yi S., editor, *Music, Mind and Science*. Seoul National University Press, 1999.

[DGW02] Simon Dixon, Werner Goebl, and Gerhard Widmer. Real Time Tracking and Visualisation of Musical Expression. In *Proc. International Conference on Music and Artificial Intelligence*, 2002.

[DGW04a] S. Dixon, F. Gouyon, and G. Widmer. Towards characterisation of music via rhythmic patterns. In *5th International Conference on Music Information Retrieval*, pages 509–516, 2004.

[DGW04b] S. Dixon, F. Gouyon, and Gerhard Widmer. Towards characterisation of music vian rhythmic patterns. In *Proceedings of the 5th ISMIR, Barcelona, Spain.*, 2004.

[DH91] P. Desain and H. Honing. The quantization of musical time: a connectionist approach. In Desain P. and Honing H., editors, *Music and Connectionism*. MIT Press, 1991.

[Dix01a] S. Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30(1):39–58, 2001.

[Dix01b] S. Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30(1):39–58, 2001.

[Dix01c] S. Dixon. An interactive beat tracking and visualisation system. In *Proceedings of the International Computer Music Conference*, pages 215–218, 2001.

[Dix01d] Simon Dixon. An interactive beat tracking and visualisation system. In *Proc. International Computer Music Conference*, 2001.

[Dix05a] S. Dixon. Live tracking of musical performances using on-line time warping. In *Proceedings of the 8th International Conference on Digital Audio Effects*, 2005. under review.

[Dix05b] S. Dixon. An on-line time warping algorithm for tracking musical performances. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2005. to appear.

[DJB00] C. Drake, M. Jones, and C. Baruch. The development of rhythmic attending in auditory sequences: attunement, referent period, focal attending. *Cognition*, 77:251–288, 2000.

[DM04] M. Davies and N Mitianoudis. Simple mixture model for sparse overcomplete ica. *IEE Proceedings on Vision, Image and Signal Processing*, 151(1):35–43, February 2004.

[DP04] M. E. P. Davies and M. D. Plumbley. Causal tempo tracking of audio. In *5th International Symposium on Music Information Retrieval*, October 2004.

[DP05a] M. E. P. Davies and M. D. Plumbley. Beat tracking with a two state model. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Philadelphia, USA, March 18–23 2005.

[DP05b] M. E. P. Davies and M. D. Plumbley. Beat tracking with a two state model. In *Proceedings of the 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Philadelphia, Penn., USA*, 2005.

[DPB00] C. Drake, A. Penel, and E. Bigand. Why musicians tap slower than nonmusicians. In Drake C., Penel A., and Bigand E., editors, *Rhythm Perception and Production*. Swets and Zeitlinger, 2000.

[DPW03] S. Dixon, E. Pampalk, and G. Widmer. Classification of dance music by periodicity patterns. In *4th International Conference on Music Information Retrieval*, pages 159–165, 2003.

[Dra93] C. Drake. Reproduction of musical rhythms by children, adult musicians and adult non-musicians. *Perception and Psychophysics*, 53(1), 1993.

[DSD02] C. Duxbury, M. Sandler, and M. Davies. A hybrid approach to musical note onset detection. In *Proceedings of the 5th International Conference in Digital Audio Effects (DAFX,'02), Hamburg, Germany*, 2002.

[DU04] C. Dittmar and C. Uhle. Further steps towards drum transcription of polyphonic music. In *Proceedings of AES 116th Convention*, Berlin, Germany, 2004.

[Dux04] Christopher Duxbury. *Signal Models for Polyphonic Music*. London, UK, June 2004.

[DW05] S. Dixon and G. Widmer. MATCH: A music alignment tool chest. In *6th International Conference on Music Information Retrieval*, 2005. under review.

[Ell96] D. P. W. Ellis. *Prediction-Driven Computational Auditory Scene Analysis*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, June 1996.

[EN00] T. Eerola and A. C. North. Cognitive complexity and the structure of musical patterns. In *Proceedings of the 6th International Conference on Music Perception and Cognition*, Newcastle, UK, 2000.

[ERD04] S. Essid, G. Richard, and B. David. Musical instrument recognition based on class pairwise feature selection. *Proc. ISMIR*, 2004.

[FCL02a] D. FitzGerald, E. Coyle, and B. Lawlor. Sub-band independent subspace analysis for drum transcription. In *Proc. of the 5th International Conference on Digital Audio Effects (DAFX-02)*, pages 65–69, 2002.

[FCL02b] Derry FitzGerald, Eugene Coyle, and Bob Lawlor. Sub-band independent subspace analysis for drum transcription. In *Proceedings of the. Digital Audio Effects Conference (DAFX02)*, 2002.

[FCL03] D. FitzGerald, E. Coyle, and B. Lawlor. Prior subspace analysis for drum transcription. In *Proc. of the 114^{th} AES convention, Amsterdam, The Netherlands*, 2003.

[FLC03] D. FitzGerald, B. Lawlor, and E. Coyle. Drum transcription in the presence of pitched instruments using prior subspace analysis. In *Proc. of the Irish Signals and Systems Conference, Limerick, Ireland*, 2003.

[Foo97] J. Foote. Content-based retrieval of music and audio. In *Multimedia Storage and Archiving Systems II*, pages 138–147, 1997.

[Foo99] Jonathan Foote. An overview of audio information retrieval. *ACM Multimedia Systems*, 7(1):2–11, 1999. ACM Press/Springer Verlag.

[Foo00] Jonathan Foote. Arthur: Retrieving orchestral music by long-term structure. In *Proceedings of the 1st International Symposium for Music Information Retrieval (ISMIR)*, Plymouth, Massachusetts, October 2000. See http://ciir.cs.umass.edu/music2000.

[FS96] Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, 1996.

[FS99] A. Friberg and J. Sundstrm. Jazz drummers' swing ratio in relation to tempo. Proc. Acoustical Society of America ASA/EAA/DAGA Meeting Lay Language Papers, 1999.

[FS02] A. Friberg and J. Sundstrm. Swing ratios and ensemble timing in jazz performances: Evidence for a common rhythmic pattern. *Music Perception*, 19(3), 2002.

[Gab73] A. Gabrielsson. Similarity ratings and dimension analyses of auditory rhythm patterns. part i. *Scandanavian Journal of Psychology*, (14):138–160, 1973.

[GD05] F. Gouyon and S. Dixon. A review of automatic rhythm description systems. *Computer Music Journal*, 29(1), 2005.

[GDPW04a] F. Gouyon, S. Dixon, E. Pampalk, and G. Widmer. Evaluating rhythmic descriptors for musical genre classification. In *Proceedings of 25th International AES Conference*, London, UK, 2004.

[GDPW04b] F. Gouyon, S. Dixon, E. Pampalk, and G. Widmer. Evaluating rhythmic descriptors for musical genre classification. In *Proceedings of the AES 25th International Conference*, pages 196–204, 2004.

[GH01a] F. Gouyon and P. Herrera. Exploration of techniques for automatic labeling of audio drum tracks instruments. In *Proceedings of MOSART Workshop on Current Research Directions in Computer Music*, Barcelona, Spain, 2001.

[GH01b] Fabien Gouyon and Perfecto Herrera. Exploration of techniques for automatic labeling of audio drum tracks instruments. In *Proc. MOSART Workshop on Current Research Directions in Computer Music*, 2001.

[GH03] Fabien Gouyon and Perfecto Herrera. A beat induction method for musical audio signals. In *Proc. WIAMIS Special session on Audio Segmentation and Digital Music*, 2003.

[GH04a] E. Gomez and P. Herrera. Automatic extraction of tonal metadata from polyphonic audio recordings. In *Proceedings of 25th International AES Conference*, London, UK, 2004.

[GH04b] E. Gomez and P. Herrera. Estimating the tonality of polyphonic audio files: Cognitive versus machine learning modelling strategies. In *Proceedings of the 5th ISMIR, Barcelona, Spain.*, 2004.

[GH04c] Emilia Gomez and Perfecto Herrera. Automatic extraction of tonal metadata from polyphonic audio recordings. In *Proceedings of the AES 25th International Conference*, pages 74–80, London, UK, 2004.

[GM94] M. Goto and Y. Muraoka. A sound source separation system for percussion instruments. *Transactions of the Institute of Electronics, Information and Communication Engineers D-II*, J77(5):901–911, 1994.

[GM95a] M. Goto and Y. Muraoka. A real-time beat tracking system for audio signals. In *Proceedings of the International Computer Music Conference*, pages 171–174, San Francisco CA, 1995. International Computer Music Association.

[GM95b] M. Goto and Y. Muraoka. A real-time beat tracking system for audio signals. In *International Computer Music Conference*, pages 171–174, 1995.

[GM97a] M. Goto and Y. Muraoka. Issues in evaluating beat tracking systems. In *Working Notes of the IJCAI-97 Workshop on Issues in AI and Music - Evaluation and Assessment*, 1997.

[GM97b] Masataka Goto and Yoichi Muraoka. Issues in evaluating beat tracking systems. In *Proc. International Joint Conference on Artificial Intelligence*, 1997.

[GM99a] M. Goto and Y. Muraoka. Real-time beat tracking for drumless audio signals. *Speech Communication*, 27(3–4):311–335, 1999.

[GM99b] Masataka Goto and Yoichi Muraoka. Real-time Beat Tracking for Drumless Audio Signals: Chord Change Detection for Musical Decisions. *Speech Communication*, 27(3), 1999.

[GM03] Fabien Gouyon and Benoit Meudic. Towards Rhythmic Content Processing of Musical Signals: Fostering Complementary Approaches. *Journal of New Music Research*, 32(1), 2003.

[Goe01] W. Goebl. Melody lead in piano performance: Expressive device or artifact? *Journal of the Acoustical Society of America*, 110(1):563–572, 2001.

[Got01] M. Goto. An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30:159–171, June 2001.

[GPD00a] F. Gouyon, F. Pachet, and O. Delerue. On the use of zero-crossing rate for an application of classification of percussive sounds. Proc. Digital Audio Effects conference, 2000.

[GPD00b] F. Gouyon, F. Pachet, and O. Delerue. On the use of zero-crossing rate for an application of classification of percussive sounds. In *Proceedings of COST G6 Conference on Digital Audio Effects 2000*, Verona, Italy, 2000.

[GPD00c] F. Gouyon, F. Pachet, and O. Delerue. On the use of zero-crossing rate for an application of classification of percussive sounds. In *Proc. of the Conference on Digital Audio Effects*, 2000.

[GTM93] M. Goto, M. Tabuchi, and Y. Muraoka. An automatic transcription system for percussion instruments. In *Proc. 46th Annual Convention IPS Japan*, 1993.

[Hai04] S. Hainsworth. *Techniques for the Automated Analysis of Musical Audio*. PhD thesis, Cambridge University, April 2004.

[Hal00] M. A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proc. of the Seventeenth International Conference on Machine Learning*, 2000.

[HDG03a] P. Herrera, A. Dehamel, and F. Gouyon. Automatic labeling of percussion sounds. In *Proceedings of the* 114^{th} *AES convention, Amsterdam, The Netherlands*, 2003.

[HDG03b] P. Herrera, A. Dehamel, and F. Gouyon. Automatic labeling of unpitched percussion sounds. In *Presented at the 114th Convention of the Audio Engineering Society*, Amsterdam, Netherlands, 2003.

[Hey75] Ronald G. Heyduk. Rated preference for musical compositions as it relates to complexity and exposure frequency. *Perception and Psychophysics*, 17(1):84–91, 1975.

[HK02] T. Heittola and A. Klapuri. Locating segments with drums in music signals. Technical report, Audio Research Group, Tampere University of Technology, Tampere, Finland, August 2002.

[Hon01] H. Honing. From time to time: The representation of timing and tempo. *Computer Music Journal*, 25(3):50–61, 2001.

[HPD03a] P. Herrera, G. Peeters, and S. Dubnov. Automatic classification of musical instrument sounds. *Journal of New Music Research*, 32(1):3–22, 2003.

[HPD03b] P. Herrera, G. Peeters, and S. Dubnov. Automatic classification of musical instrument sounds. *Journal of New Music Research*, 32(1), 2003.

[HS05] Christopher A. Harte and Mark B. Sandler. Automatic chord identification using a quantised chromagram. In *Proceedings of the 118th Convention of the Audio Engineering Society*, Barcelona, Spain, May 28-31 2005.

[HSG04] P. Herrera, V. Sandvold, and F. Gouyon. Percussion-related semantic descriptors of music audio files. In *Proceedings of 25th International AES Conference*, London, UK, 2004.

[HSGP05] P. Herrera, V. Sandvold, F. Gouyon, and E. Pampalk. Semantic interaction with music audio contents using percussion-related descriptors. *Journal of Intelligent Information Systems*, 2005. (accepted).

[Ita75] F. Itakura. Line spectrum representation of linear predictive coefficients of speech signals. *J. Acoust. Soc. Amer.*, 57:S35, 1975.

[JB89] M. Jones and M. Boltz. Dynamic attending and responses to time. *Psychological Review*, 96(3), 1989.

[JCB92] Miller S. Puckette Judith C. Brown. An efficient algorithm for th calculation of a constant q transform. *Journal of the Acoustical Society of America*, 92(5):2698–2701, 1992.

[JIM+04] Heather D. Jennings, Plamen Ch. Ivanov, Allan M. Martins, P. C. da Silva, and G. M. Viswanathan. Variance fluctuations in nonstationary time series: a comparative study of music genres. *Physica A: Statistical and Theoretical Physics*, 336(3-4):585–594, May 2004.

[Jør] M. Jørgensen. Drumfinder: Dsp project on recognition of drum sounds in drum tracks. http://www.daimi.au.dk/∼elmer/dsp.

[Kau02] Ismo Kauppinen. Methods for detecting impulsive noise in speech and audio signals. In *Proceedings of DSP-2002*, July 2002.

[Kla99] A. Klapuri. Sound Onset Detection by Applying Psychoacoustic Knowledge. In *Proc. IEEE Conf. Acoustics, Speech and Signal Proceesing (ICASSP,'99)*, 1999.

[Kla03a] A. Klapuri. Musical meter estimation and music transcription. Proc. Cambridge Music Processing Colloquium, 2003.

[Kla03b] A. P. Klapuri. Musical meter estimation and transcription. In *Cambridge Music Signal Processing Colloquium*, 2003.

[Kla03c] Anssi Klapuri. Musical meter estimation and music transcription. In *Proc. Cambridge Music Processing Colloquium*, 2003.

[KR86] P. Kabal and R.P. Ramachandran. The computation of line spectral frequencies using chebyshev polynomials. *IEEE trans. on Acoustics, Speech and Signal Processing*, ASSP 34(6):1419–1426, 1986.

[Kra01] M. Kragtwijk. Percussive music and the automatic generation of 3d animations. M.sc. thesis, University of Twente, The Netherlands, 2001.

[Kru90] C. L. Krumhansl. *Cognitive Foundations of Musical Pitch*. Oxford University Press, New York, 1990.

[KS04] A.G. Krishna and T.V. Sreenivas. Music instrument recognition: from isolated notes to solo phrases. *Proc. ICASSP*, pages 265–268, 2004.

[KVES01] A. Klapuri, T. Virtanen, A. Eronen, and J. Seppé4nen. Automatic transcription of musical recordings. In *Consistent and Reliable Acoustic Cues Workshop, CRAC-01*, September 2001.

[Lap00] E. Lapidaki. Stability of tempo perception in music listening. *Music Education Research*, 2(1), 2000.

[LBG80] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:702–710, 1980.

[LJ83a] F. Lerdahl and R. Jackendoff. *A generative theory of tonal music*. MIT Press, Cambridge MA USA, 1983.

[LJ83b] Fred Lerdahl and Ray Jackendoff, editors. *A generative theory of tonal music*. MIT Press, Cambridge MA, 1983.

[LM81] Henry Ledgard and Michael Marcotty. *The Programming Language Landscape*. Science Research Associates, Inc., Chicago, 1981.

[Mar98] K.D. Martin. Musical instrument identification: a pattern recognition approach. *Presented at the 136th meeting of the Acoustical Society of America*, 1998.

[Mas96] P. Masri. *Computer Modeling of Sound for Transformation and Synthesis of Musical Signals*. PhD Thesis, University of Bristol, 1996.

[MB03] M.F. McKinney and J. Breebaart. Features for audio and music classification. In *4th International Conference on Music Information Retrieval*, pages 151–158, 2003.

[McL92] G.J. McLachlan. *Discriminant analysis and statistical pattern recognition*. John Wiley and Sons Inc., 1992.

[MEV98] K. D. Martin, E.D.Scheirer, and B.L. Vercoe. Music content analysis through models of audition. In *Proc. of the ACM Multimedia Workshop on Content Processing of Music for Multimedia Applications, Bristol, UK*, 1998.

[MG96] B. C. J. Moore and B. R. Glasberg. A revision of Zwicker's loudness model. *Acustica – Acta Acustica*, 82:335–345, 1996.

[MM99] J. Marques and P. J. Moreno. A study of musical instrument classification using gaussian mixtures models and support vector machines. *Compaq Cambridge Research Laboratory*, Tech. Report 99-4, 1999.

[Moe02] D. Moelants. Preferred tempo reconsidered. Proc. of the International Conference on Music Perception and Cognition, 2002.

[MQ86] R.J. McAulay and T.F. Quatieri. Speech analysis/synthesis based on a sinusoidal representation. *IEEE Transactions in Acoustics, Speech and Signal Processing*, ASSP-34:744–754, 1986.

[MR97] Dirk Moelants and Christian Rampazzo. *KANSEI - The Technology of Emotion*, chapter A Computer System for the Automatic Detection of Perceptual Onsets in a Musical Signal, pages 141–146. Genova: AIMI-DIST, 1997.

[MXKS04] N. C. Maddage, C. Xu, M. S. Kankanhalli, and X. Shao. Content-based music structure analysis with the applications to music semantic understanding. In *ACM Multimedia Conference*, pages 112–119, 2004.

[NH97] Adrin C. North and David J. Hargreaves. Liking, arousal potential, and emotions expressed by music. *Scandinavian Journal of Psychology*, 38:45–53, 1997.

[OH04] B. S. Ong and P. Herrera. Computing structural descriptions of music through the identification of representative excerpts from audio files. In *Proceedings of 25th International AES Conference London*, 2004.

[oI] The University of Iowa. Musical instrument database. *http://theremin.music.uiowa.edu*.

[Ori01] I. Orife. Riddim: A rhythm analysis and decomposition tool based on independent subspace analysis. Master of arts thesis, Dartmouth College, Hanover, NH, 2001.

[Pal97] C. Palmer. Music performance. *Annual review of psychology*, 48:115–138, 1997.

[Pam01] E. Pampalk. Islands of music: Analysis, organization, and visualization of music archives. Master's thesis, Vienna University of Technology, Department of Software Technology and Interactive Systems, 2001.

[Pam04] E. Pampalk. A matlab toolbox to compute music similarity from audio. In *Proceedings of the Fifth International Conference on Music Information Retrieval (ISMIR'04)*, Barcelona, Spain, October 10-14 2004.

[Par04] R. M. Parry. Musical complexity and top 40 chart performance. Technical Report. College of Computing, Georgia Institute of Technology, 2004.

[PB02] Bryan Pardo and William P. Birmingham. Algorithms for chordal analysis. *Computer Music Journal*, 26(2):27–49, 2002.

[PBH99] Jan Puzicha, Joachim M. Buhmann, and Thomas Hofmann. Histogram clustering for unsupervised image segmentation. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Ft. Collins, CO, USA*, pages 2602–2608, 1999.

[PBM+02] Jeremy Pickens, Juan Pablo Bello, Giuliano Monti, Tim Crawford, Matthew Dovey, Mark Sandler, and Don Byrd. Polyphonic score retrieval using polyphonic audio queries: A harmonic modeling approach. In *Proceedings of the 3rd Annual International Conference on Music Information Retrieval (ISMIR)*, pages 140–149, Paris, France, October 2002.

[PBO00] Hendrik Purwins, Benjamin Blankertz, and Klaus Obermayer. A new method for tracking modulations in tonal music in audio data format. In *International Joint Conference on Neural Networks (IJCNN'00)*, volume 6, pages 270–275, 2000.

[PBR02] G. Peeters, A. La Burthe, and X. Rodet. Toward automatic music audio summary generation from signal analysis. In *Proceedings of ISMIR 2002, 3rd International Conference on Music Information Retrieval*, pages 94–100, October 2002.

[PDW04] E. Pampalk, S. Dixon, and G. Widmer. Exploring music collections by browsing different views. *Computer Music Journal*, 28(2):49–62, 2004.

[Ped01] T. H. Pedersen. Objective method for measuring the prominence of impulsive sounds and for adjustment of laeq. In *Proceedings of the International Congress and Exhibition on Noise Control, Den Haag, The Netherlands*, 2001.

[Pee04] G. Peeters. A large set of audio features for sound description (similarity and classification) in the CUIDADO project. Cuidado i.s.t. project report, 2004.

[Pen] C.-K. Peng. Fractal mechanics in neural control: Human heartbeat and gait dynamics in health and disease. http://www.physionet.org/tutorials/fmnc/. Online Tutorial.

[PFW05] E. Pampalk, A. Flexer, and G. Widmer. Improvements of audio-based music similarity and genre classification. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2005. submitted.

[PGW03] Elias Pampalk, Werner Goebl, and Gerhard Widmer. Visualizing changes in the structure of data for exploratory feature selection. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington DC, August 24-27 2003. ACM.

[PK02] J. Paulus and A. Klapuri. Measuring the similarity of rhythmic patterns. In *Proceedings of the 3rd International Conference on Musical Information Retrieval*, pages 150–156. IRCAM Centre Pompidou, 2002.

[PK03a] J. Paulus and A. Klapuri. Conventional and periodic n-grams in the transcription of drum sequences. In *Proc. of IEEE International Conference on Multimedia and Expo, Baltimore, USA*, 2003.

[PK03b] J. Paulus and A. Klapuri. Model-based event labeling in the transcription of percussive audio signals. In *Proceedings of the 6th International Conference on Digital Audio Effects*, London, UK, 2003.

[PK03c] J. Paulus and A. Klapuri. Model-based event labelling in the transcription of percussive audio signals. In *Proceedings of the 6^{th} Conference on Digital Audio Effects, London, UK*, 2003.

[PK03d] Jouni Paulus and Anssi Klapuri. Model-based Event Labeling in the Transcription of Percussive Audio Signals. In *Proc. International Conference on Digital Audio Effects*, 2003.

[PRM02] Elias Pampalk, Andreas Rauber, and Dieter Merkl. Content-based organization and visualization of music archives. In *Proceedings of the ACM Multimedia*, pages 570–579, Juan les Pins, France, December 1-6 2002. ACM.

[Rab89] L. R. Rabiner. A tutorial on HMM and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[RC99] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, New York, 1999.

[Rep92] B. Repp. Probing the cognitive representation of musical time: structural constraints on the perception of timing perturbations. *Cognition*, 44:241–281, 1992.

[Rep94] B.H. Repp. On determining the basic tempo of an expressive music performance. *Psychology of Music*, 22:157–167, 1994.

[Ris02] E. Riskedal. Drum analysis. M.sc. thesis, Dept. of Informatics, Univ. Bergen, Norway, 2002.

[RJ01] Xavier Rodet and Florent Jaillet. Detection and modeling of fast attack transients. In *Proceedings of the International Computer Music Conference*, 2001.

[RS03] Christopher Raphael and Josh Stoddard. Harmonic analysis with probabilistic graphical models. In *Proceedings of the 4th Annual International Conference on Music Information Retrieval (ISMIR)*, pages 177–181, Baltimore, Maryland, October 2003.

[Sau96] J. Saunders. Real time discrimination of broadcast speech/music. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 993–996, 1996.

[SB00] Kåre Sjölander and Jonas Beskow. WaveSurfer - An open source speech tool. In *Proc. International Conference on Spoken Language Processing*, 2000.

[Sch98a] E. D. Scheirer. Tempo and beat analysis of acoustic musical signals. *Journal of Acoustical Society of America*, 103:588–601, January 1998.

[Sch98b] E.D. Scheirer. Tempo and beat analysis of acoustic musical signals. *Journal of the Acoustical Society of America*, 103(1):588–601, 1998.

[Sch99] R. E. Schapire. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

[Sch00] E.D. Scheirer. *Music-Listening Systems*. PhD thesis, Massachusetts Institute of Technology, School of Architecture and Planning, 2000.

[SE03a] Alexander Sheh and Daniel P. W. Ellis. Chord segmentation and recognition using em-trained hidden markov models. In *Proceedings of the 4th Annual International Conference on Music Information Retrieval (ISMIR)*, pages 183–189, Baltimore, Maryland, October 2003.

[SE03b] Alexander Sheh and Daniel P.W. Ellis. Chord segmentation and recognition using em-trained hidden markov models. *Proceedings of the ICMC 2003*, 2003.

[Ser89] X. Serra. *"A System for Sound Analysis/Transformation/Synthesis Based on a Deterministic plus Stochastic Decomposition."*. PhD Diss., Stanford University, 1989.

[SGH04] V. Sandvold, F. Gouyon, and P. Herrera. Drum sound classification in polyphonic audio recordings using localized sound models. In *Proceedings of Fifth International Conference on Music Information Retrieval*, Barcelona, 2004.

[SH05] Sebastian Streich and Perfecto Herrera. Detrended fluctuation analysis of music signals: Danceability estimation and further semantic characterization. In *Proceedings of the 118th Convention of the Audio Engineering Society*, Barcelona, Spain, 2005.

[She64] Roger Shepard. Circularity in judgments of relative pitch. *Journal of the Acoustical Society of America*, 35:2346–2353, 1964.

[SN04] E. Skovenborg and S. Nielsen. Evaluation of different loudness models with music and speech material. In *Proceedings of the AES 117th Convention*, San Francisco, CA, USA, 2004.

[Sou] G.A. Soulodre. Evaluation of objective loudness meters. In *Proceedings of the AES 116th Convention*, Berlin, Germany.

[Tag03] Philip Tagg. Tagg's harmony handout, available at www.tagg.org. *Version 3, (accessed 21/04/05)*, 2003.

[Tan93] Andranick S. Tanguiane. *Artificial Intelligence and Music Recognition.* Number 746 in Lecture Notes in Artificial Intelligence. Springer-Verlag, 1993.

[Tay89] Eric Taylor. *The AB Guide to Music Theory Part 1.* ABRSM Publishing Ltd, Portland Place, London, UK, 1989.

[TC99] G. Tzanetakis and P. Cook. Multifeature audio segmentation for browsing and annotation. In *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 103–106, October 1999.

[TC02] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[TEC01] G. Tzanetakis, G. Essl, and P. Cook. Automatic musical genre classification of audio signals. In *International Symposium on Music Information Retrieval*, 2001.

[UD04] Christian Uhle and Christian Dittmar. Generation of Musical Scores of Percussive Unpitched Instruments from Automatically Detected Events. In *Proc. 116th AES Convention*, 2004.

[UDS03] C. Uhle, C. Dittmar, and T. Sporer. Extraction of drum tracks from polyphonic music using independent subspace analysis. In *Proceedings of the International Conference on Independent Components Analysis, Nara, Japan*, 2003.

[Vic01] Earl Vickers. Automatic long-term loudness and dynamics matching. In *Proceedings of the 111th Convention of the Audio Engineering Society*, New York, 2001.

[Vir03] T. Virtanen. Sound source separation using sparse coding with temporal continuity objective. In *Proceedings of the International Computer Music Conference, Singapore*, 2003.

[vNM99] L. van Noorden and D. Moelants. Resonance in the perception of musical pulse. *Journal of New Music Research*, 28(1):43–66, March 1999.

[VR81] J. Vos and R. Rasch. The perceptual onset of musical tones. *Perception and Psychophysics*, 29(4):323–335, 1981.

[VR04] Emmanuel Vincent and Xavier Rodet. Underdetermined source separation with structured source priors. In C. Puntonet and A. Prieto, editors, *Independent Component Analysis and Blind Signal Separation: Fifth International Conference, ICA 2004, Granada, Spain, September 22–24, 2004*, volume 3195 of *Lecture Notes in Computer Science*, pages 327–334. Springer-Verlag, Heidelberg, Germany, 2004.

[vSTD+04] D. van Steelant, K. Tanghe, S. Degroeve, B. De Baets, M. Leman, and J.-P. Martens. Classification of percussive sounds using support vector machines. In *Proceedings of the annual machine learning conference of Belgium and The Netherlands, Brussels, Belgium*, 2004.

[WBKW96] E. Wold, T. Blum, D. Keislar, and J. Wheaton. Content-based classification, search, and retrieval of audio. *IEEE Multimedia*, 3(2):7–36, 1996.

[Wei03] C. Wei. Structural analysis of musical signals for indexing, segmentation and thumbnailing. *Paper for the Major Area of the PhD General Exam*, 2003.

[WF99] I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, CA, 1999.

[WS05] X. Wen and M. Sandler. A partial searching algorithm and its application for polyphonic music transcription. Accepted for poster presentation at the 6th ISMIR Conference, London, UK, 2005.

[WTAV03] Y. Wang, J. Tang, A. Ahmaniemi, and M. Vaalgamaa. Parametric vector quantization for coding percussive sounds in music. In *Proceedings of the IEEE International Conference on Speech and Signal Processing, Hong Kong*, 2003.

[Yan02] C. Yang. MACSIS: A scalable acoustic index for content-based music retrieval. In *Proceedings of the 3rd ISMIR, Paris, France.*, 2002.

[YR04a] O. Yilmaz and S. Rickard. Blind separation of speech mixtures via time-frequency masking. *IEEE Transactions on Signal Processing*, 52(7):1830–1847, July 2004.

[YR04b] Özgür Yılmaz and Scott Rickard. Blind separation of speech mixtures via time-frequency masking. *IEEE Transactions on Signal Processing*, 52(7):1830–1847, July 2004.

[ZF90] E. Zwicker and H. Fastl. *Psychoacoustics - Facts and Models*. Springer-Verlag, 1990.

[ZP01] Michael Zibulevsky and Barak A. Pearlmutter. Blind source separation by sparse decomposition in a signal dictionary. *Neural Computation*, 13(4):863–882, 2001.

[ZP03] A. Zils and F. Pachet. Extracting automatically the perceived intensity of music titles. In *Proceedings of the 6th Int. Conference of Digital Audio Effects (DAFX-03)*, London, UK, 2003.

[ZPDF02] A. Zils, F. Pachet, O. Delerue, and F.Gouyon. Automatic extraction of drum tracks from polyphonic music signals. In *2nd International Conference on Web Delivering of Music (WedelMusic)*, 2002.

[ZPDG02] A. Zils, F. Pachet, O. Delerue, and F. Gouyon. Automatic extraction of drum tracks from polyphonic music signals. In *Proceedings of International Conference on Web Delivering of Music 2002*, Darmstadt, Germany, 2002.