TAMPERE UNIVERSITY OF TECHNOLOGY
Department of Information Technology

Matti Ryynänen

Probabilistic Modelling of Note Events in the Transcription of Monophonic Melodies

Master of Science Thesis

The subject was approved by the Department of Information Technology on the 11th of February 2004
Thesis supervisors: Prof. Jaakko Astola
MSc Anssi Klapuri

Preface

This work was carried out at the Institute of Signal Processing, Tampere University of Technology.

I want to thank my examiner, Professor Jaakko Astola, for his advice and comments.

In particular, I wish to express my humblest compliments to MSc Anssi Klapuri who has patiently guided me throughout this work, answered to my questions, and provided me the opportunity to work in the Audio Research Group. Without his advice, this work would have required twice as much coffee.

In addition, I would like to thank MSc Timo Viitaniemi for his cooperation in the melody transcription research, MSc Antti Eronen for his comments and help, and the personnel of the Audio Research Group. It has been a pleasure to work with you.

I am very grateful to my parents, Pirkko and Reino, who have always encouraged and loved me. I love you very much. And there are friends who have helped me not to lose my marbles, provided me memorable moments, and been there for me. Thank you, you know what you are to me.

And finally during this work, I found the answer to Mr Frank Zappa's question, does humour belong in music? The answer is — yes, it must.

Tampere, March 2004

Matti Ryynänen
Makasiininkatu 14 B 32
33230 Tampere
Finland
tel: +358 50 5360 693
e-mail: `matti.ryynanen@tut.fi`

Tiivistelmä

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan osasto
Signaalinkäsittelyn laitos

RYYNÄNEN, MATTI: Nuottitapahtumien tilastollinen mallintaminen yksiäänisen melodian nuotintamisessa
Diplomityö, 80 sivua
Tarkastajat: Prof. Jaakko Astola, DI Anssi Klapuri
Rahoittaja: Nokia Research Center
Maaliskuu 2004
Avainsanat: musiikin sisältöanalyysi, laulun nuotintaminen, tilastollinen mallintaminen, musiikkitieteellinen mallintaminen, kätketty Markovin malli

Tämä diplomityö käsittelee musiikin automaattista nuotintamista ja esittää menetelmän yksiäänisten melodioiden nuotintamiseksi. Musiikin sisällön analysoiminen tietokoneiden avulla on viime vuosina herättänyt suurta mielenkiintoa tutkijoiden keskuudessa, mihin eräänä syynä on ollut digitaalisten musiikkitietokantojen räjähdysmäinen kasvu. Melodioiden automaattinen nuotintaminen on eräs tämän tutkimusalueen keskeisistä ongelmista, jonka ratkaisemisesta olisi iloa niin musiikin ammattilaisille kuin tavallisille kuluttajillekin. Musiikin nuotintamisen lisäksi mahdollisia sovelluksia ovat esimerkiksi musiikkikappaleiden haku digitaalisista musiikkitietokannoista lauletun melodian perusteella ja vuorovaikutteiset musiikkisovellukset.

Työssä esitetty menetelmä perustuu kahteen tilastolliseen malliin, joista ensimmäinen mallintaa yksittäisiä nuottitapahtumia ja toinen näiden välisiä siirtymiä. Tilastollinen malli nuottitapahtumalle rakennetaan käyttäen akustista tietokantaa ja kätkettyä Markovin mallia. Nuottitapahtumien välisten siirtymien mallintamisessa hyödynnetään puolestaan sävellajin arviointia ja nuottisekvenssien todennäköisyyksiä. Näiden mallien avulla muodostetaan modulaarinen järjestelmä, jolla voidaan nuotintaa melodioita.

Työssä esitetyn järjestelmän toimivuus on arvioitu ja tulokset ovat hyviä. Järjestelmä kykenee nuotintamaan yli 90 prosenttia nuoteista oikein, mikä tarkoittaa virheiden määrän vähentymistä puoleen verrattuna yksinkertaiseen menetelmään, jossa äänenkorkeusmittaukset pyöristetään lähimpiin nuotteihin. Erityisesti nuottitapahtumien mallintaminen lisää järjestelmän suorituskykyä.

Abstract

TAMPERE UNIVERSITY OF TECHNOLOGY
Department of Information Technology
Institute of Signal Processing

RYYNÄNEN, MATTI: Probabilistic Modelling of Note Events in the Transcription of Monophonic Melodies
Master of Science Thesis, 80 pages
Examiners: Prof. Jaakko Astola, MSc Anssi Klapuri
Funding: Nokia Research Center
March 2004
Keywords: music content analysis, music transcription, statistical modelling, musicological modelling, hidden Markov model

This thesis concerns the problem of automatic transcription of music and proposes a method for transcribing monophonic melodies. Recently, computational music content analysis has received considerable attention among researchers due to the rapid growth of music databases. In this area of research, melody transcription plays an important role. Automatic melody transcription can benefit professional musicians and provide interesting applications for consumers, including music retrieval and interactive music applications.

The proposed method is based on two probabilistic models: a note event model and a musicological model. The note event model is used to represent note candidates in melodies, and it is based on hidden Markov models. The note event model is trained with an acoustic database of singing sequences, but the training can be done for any melodic instrument and for the desired front-end feature extractors. The musicological model is used to control the transitions between the note candidates by using musical key estimation and by computing the likelihoods of note sequences. The two models are combined to constitute a melody transcription system with a modular architecture.

The transcription system is evaluated, and the results are good. Our system transcribes correctly over 90 % of notes, thus halving the amount of errors compared to a simple rounding of pitch estimates to the nearest MIDI note numbers. Particularly, using the note event model significantly improves the system performance.

Contents

1 Introduction

Transcription of music is a process of generating symbolic notations, i.e., *musical transcriptions*, from musical performances. Conventionally, musical transcriptions have been written by hand, requiring both time and musical education. If music transcription could be automatically accomplished by computers, it would significantly benefit music professionals and, more importantly, provide an opportunity for common people to interact with music. In addition, the rapid growth of digital music databases has challenged researchers to develop tools for the automatic analysis and indexing of musical information, thus establishing an interesting topic of research: computational *music content analysis*. The automated analysis of music enables several applications besides the automatic transcription of music, including music retrieval, object-based coding of music, and interactive music applications. In addition, automatic music content analysis provides a natural way for users to interact with digital music, for example, by singing, humming, or clapping hands.

Music content analysis is largely based on *melodies* which are note sequences with organised and recognisable shape. Since melodies enable us to distinguish one musical excerpt from another [Selfridge-Field98], several solutions have been suggested for extracting melodies from music signals. These solutions are then used in *melody transcription systems*, which produce note sequences from acoustic music inputs. Figure 1.1 illustrates the melody transcription problem.

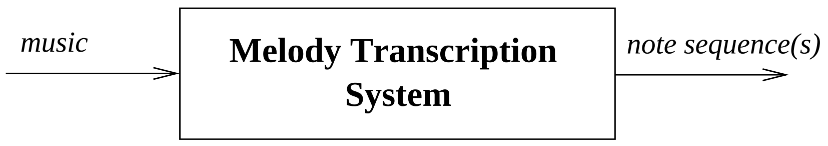

Figure 1.1: An illustration of the melody transcription problem.

The state-of-the-art melody transcription systems are focused on transcribing singing sequences [Clarisse02, Viitaniemi03, Pollastri02], and they typically exploit computational models of hearing and musicological knowledge. However, these solutions are based on short time-frames or preprocessed segments of audio signals, thus ignoring the importance of notes as musicological units having dynamic nature.

This thesis proposes a statistical note event model and a musicological model that constitute a system for transcribing monophonic melodies, i.e., melodies with only a single note sounding at a time. The note event model is used to represent note candidates in music performances, and the relationships between consecutive notes are statistically examined by the musicological model. Briefly, the note event model uses *musical features* to calculate likelihoods for different note candidates. Musical features represent the musically meaningful contents of audio signals, and they are extracted by well-known methods. The musicological model, on the other hand, uses *key estimation* and likelihoods of different *note sequences* to examine transitions between note candidates. In particular, the proposed models are not bounded to particular feature extractors, and they can be extended to meet the criteria of different melody transcription applications.

This thesis focuses on applying the methods used in speech recognition to the modelling of note events and on establishing an architecture which can be extended by new musical features and musicological rules. The discovery of purely signal-processing-based algorithms is outside the scope of this thesis. Instead, the applicative research made in this work strives for an interdisciplinary approach to combine signal processing and musicology, of which the former is naturally emphasised.

Organisation of the thesis

This thesis is organised as follows. Chapter 2 reviews the literature concerning music-content analysis and focuses on the approaches and methods previously applied in melody transcription. Chapter 3 discusses the extraction of musical features from audio signals. Chapter 4 proposes the probabilistic models for melody transcription, including the note event model and the musicological model. In Chapter 5, the extraction of musical features and the probabilistic models are integrated to form a melody transcription system, and the system performance is evaluated and the results are represented in Chapter 6. Finally, Chapter 7 summarises the observations made during this work.

2 Literature Review

This chapter reviews the ideas and methods used in computational music-content analysis, particularly those applied in automatic music transcription. Music content analysis as a research area consists of ideas, tools and applications of analysing acoustic music signals. In most cases, psychoacoustics and music psychology are considered in order to mimic human-like interpretation of music signals.

This chapter is organised as follows. Section 2.1 discusses the perception of music signals in general and Section 2.2 provides a short summary of music terminology. Sections 2.3 and 2.4 review the methods used in automatic transcription of melodies and rhythm, respectively. In Section 2.5, the applications of automatic music-content analysis are surveyed.

2.1 Perception of music signals

Human listeners can easily orient to musical sounds, even without any formal musical education. The basic features of acoustic signals, such as *pitch*, *loudness*, *duration*, and *timbre*, are easily perceived and further processed to constitute musical concepts, such as melodies, rhythm, and harmony. The study of music signal perception is divided into two main approaches. *Psychoacoustics* is a branch of science which studies the relationship between an acoustic signal and the percept it evokes. *Music psychology*, on the other hand, concerns the human ability to organise musical information into perceptual structures and the listeners' affection evoked by a musical stimulus [Scheirer00, p. 17]. Perception of the most important acoustic features are reviewed in this section so that the computational analysis of musical elements is justified to an adequate extent. Although these topics are under extensive research, and especially psychoacoustics is widely exploited in music signal processing applications, there is still much to discover about the human auditory system and the neurophysiological impact caused by music signals.

2.1.1 Pitch

Perception of pitch has been extensively studied due to its fundamental significance to the human auditory system. In ANSI standard 1994 [ANSI94], pitch is described as follows:

> "Pitch is that attribute of auditory sensation in terms of which sounds may be ordered on a scale extending from low to high. Pitch depends mainly on the frequency content of the sound stimulus, but it also depends on the sound pressure and the waveform of the stimulus."

This verbal definition, however, is not sufficient for our purposes to analytically examine pitch. The operational definition of pitch associates a measurable physical quantity, frequency, with the pitch [Stevens75]:

> "A sound has a certain pitch if it can be reliably matched by adjusting the frequency of a sine wave of arbitrary amplitude."

There are two competing theories to explain how the auditory system derives a sensation of pitch from a sinusoidal signal. The *place theory* of pitch emphasises the fact that the preliminary frequency analysis of acoustic signals is made on the basilar membrane in the inner ear. The basilar membrane vibrates according to the frequencies of acoustic signals, and the physical position of the resonance peaks causes a pitch sensation. The *timing theory*, on the other hand, suggests that frequency analysis is made according to the temporal behaviour of the observed peaks on the basilar membrane. Nevertheless, neither of these theories completely explains the linkage between an acoustic input and a pitch sensation. In general, both theories are considered to operate simultaneously given that the timing theory dominates at the lower frequencies, and the place theory at the higher frequencies [Hartmann96, p. 3497].

Pitch is a subjective attribute of a sound due to its perceptual nature. The physical counterpart of pitch is the *fundamental frequency* F_0 which is defined for perfectly periodic signals as follows [Cheveigné02].

> "The fundamental frequency of a periodic signal is the inverse of its period which may be defined as the smallest positive member of the infinite set of time shifts that leave the signal invariant."

Although the terms pitch and fundamental frequency are different expressions of the same concept, we follow the common practise and use the terms as synonyms. However, the fundamental frequency is defined only for periodic or nearly periodic signals whereas a pitch may be heard even for completely non-periodic signals. For example, pitch may be assigned for the sound of a church bell, even though the sound does not have a fundamental frequency.

Pitch of complex tones

Considering more natural sounds than a simple sine wave, *complex tones* are composed of several frequency components. Periodic complex tones consist of harmonics which are multiples of the fundamental frequency, thus affecting the sensation of pitch. As an example, Figure 2.1 represents a complex tone consisting of sines

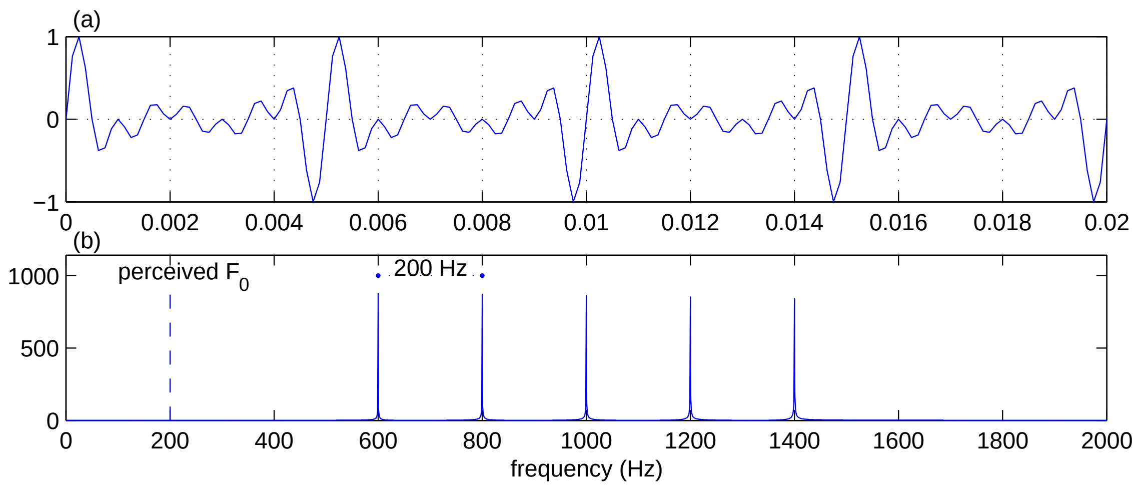

Figure 2.1: A complex tone with the fundamental frequency $F_0 = 200$ Hz; (a) the waveform and (b) the frequency content.

with frequencies 600 Hz, 800 Hz, 1000 Hz, 1200 Hz, and 1400 Hz. The sound has a subjective pitch of 200 Hz, although there exists no such frequency component. Furthermore, a subjective pitch could be sensed even if the harmonics were not in exact integral relations which is, in fact, typical for natural sounds.

The peripheral parts of the auditory system do not completely explain pitch perception. Therefore, pitch perception model called *pattern-matching* (or *template-fitting*) has been proposed. The pattern-matching model suggests what happens in the brain during the perception of sounds, apart from the physiology of hearing. Goldstein presented a theory of the *central formation* of a pitch [Goldstein73], suggesting that pitch is assigned to a best-fitting collection of aurally resolvable harmonics. In other words, the subjective pitch is derived from the frequencies of the harmonics by the pattern-matching method, and the individual harmonics are less important. Terhardt proposed in [Terhardt74] that the *pitches* of the harmonics are relevant to the pattern-matching model, not the frequencies as proposed in the Goldstein's model.

The present concept of pitch was outlined by Meddis and Hewitt in [Meddis91]. They proposed a computational pitch-perception model based on psychoacoustic studies, considering the physiological functioning of the ear, both the place theory and the timing theory of pitch perception, and the pattern-matching model. At present, this model is the most comprehensive auditory pitch model, and its practical usage in pitch extraction is further discussed in sections 2.3.1 and 2.3.2.

2.1.2 Rhythm

The perception of *rhythm* has not gained as much researchers' attention as pitch. This was until recently when an appropriate, conceptual description of the rhythm itself was proposed by Lerdahl and Jackendoff in [Lerdahl83]. By the description, rhythm consists of *grouping* and *meter* which are discussed in the following. Empirical verifications of the theory are surveyed in [Clarke99, pp. 482-489].

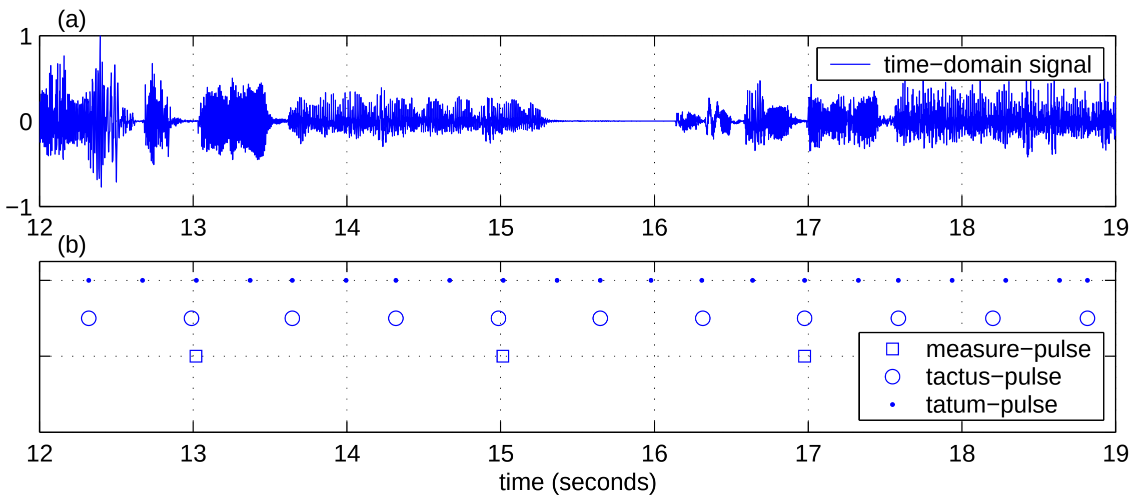

Figure 2.2: Three levels of meter: *tatum*, *tactus*, and *measure* pulses estimated from a time-domain signal.

Grouping

Grouping means hierarchical segmentation of a musical piece into variable-sized rhythmic structures. The time-scales of these perceptual structures extend from groups of a few successive notes to entities considering the whole piece. The formation of these structures is driven by the common human cognition: we tend to group together the things that are similar (the principle of similarity) and the things that are closely-spaced (the principle of proximity) [Temperley01, p. 55].

Lerdahl and Jackendoff suggest two sets of rules for the perceptual grouping of rhythm. The first set, the *Grouping Well-Formedness Rules* (GWFR), describes the form of legal structures. GWFR includes, e.g., the following rules: each group must consist of contiguous music elements, the groups must not overlap each other, and the groups containing a smaller group must be further partitioned into smaller groups. The second set of rules, the *Grouping Preference Rules* (GPR), defines the set of legal structures which are most likely heard by listeners. For example, articulation of notes or changes in dynamics suggest group boundaries.

Meter

The second element of rhythm is the meter which refers to a regular alternation of strong and weak beats sensed by a listener. The term meter induction is often used of this process to emphasise that the meter does not need to be explicitly spelled out in music, but it is inducted by observing the underlying rhythmic periodicities in music signals. A simple example of the meter induction is a listener tapping the rhythm along with a music piece. Similarly to grouping, meter has a hierarchical structure allowing metrical levels to be divided into shorter metrical levels in time. This is illustrated in Figure 2.2 where three levels of meter are represented. Figure

2.2(a) shows a time-domain signal from which the metrical levels are estimated and represented in Figure 2.2(b).

The term *tactus*, as the most important metrical level, refers to a "perceptually the most prominent level of metrical structure" which is continuous and regular throughout the piece [Lerdahl83, p. 71]. In other words, this is the most intuitive rate to tap along a music piece. Likewise for grouping, Lerdahl and Jackendoff define *Metrical Well-Formedness Rules* (MWFR) and *Metrical Preference Rules* (MPR) to explain meter perception.

2.1.3 Timbre

Pitch, loudness, duration, and timbre are commonly considered to be the perceptual dimensions of sound. While the first three attributes can be determined and defined in a consistent way, the definition of timbre remains rather vague. Superficially, timbre is the colour of sound, or from a listener's point of view, timbre is what something *sounds* like. In ANSI 1973 standard, timbre is defined as the quality of sound by which a listener can distinguish two sounds having equal pitch, duration, and loudness. Yet, this does not actually point out what timbre is, only what it is not.

Timbre is a complex concept which is affected by several physical variables, including frequency content, temporal envelope, and the context of a sound. Nevertheless, timbre may not be uniquely determined by any acoustic feature or a combination of these [Handel95]. Despite this, timbre enables the identification of sound sources by humans.

The identification of sound sources according to their timbres is explained by two approaches. The first emphasises that there are invariant physical properties or actions of a sound source which lead to the recognition of the source. For example, instruments can be identified by timbre irrespective of the music played with them. Second, timbre may be perceived due to experience and learning on similar sounds.

Various psychoacoustic experiments have been conducted to find the appropriate set of sound dimensions defining timbre. The most prominent dimensions found are the spectral centroid, the rise time, and the temporal behaviour of the harmonics of the sound [Bregman90]. The spectral centroid of a sound describes the relative amount of spectral energy on high and low frequencies, leading to a sensation of *brightness*. The rise time (or the *attack*) of a sound is the time from the sound onset to the moment when the maximal signal amplitude is achieved. The temporal behaviour of the harmonics correlates with the roughness of the sound.

2.2 Music terminology

The most fundamental units in music are *notes* which have an identifiable pitch and a duration. The note pitches corresponding to the white keys of a piano are named

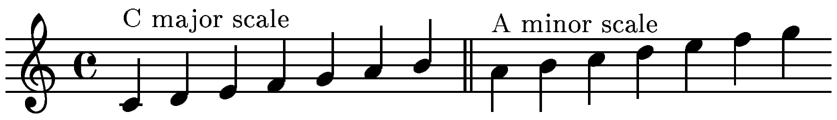

Figure 2.3: Relative C major and A minor scales.

with letters C, D, E, F, G, A, and B. Several notes together constitute *melodies* and *chords*. Melodies are consecutive sequences of notes with recognisable shapes, whereas chords are combinations of two or more simultaneous notes. The pitch difference between two notes is referred to as an *interval*. In particular, intervals with pitch frequency ratio 2:1 are called *octaves*. Each octave is divided into twelve notes in Western music. This twelve-note system defines the frequencies f of note pitches by the following equation,

$$f = 2^{\frac{x}{12}} f_{base}, \tag{2.1}$$

in which f_{base} is the frequency of a reference note (usually $f_{base} = 440$ Hz is used for the note A), and x is an integer offset from the reference note. The intervals between two adjacent note pitches are measured in *semitones*. Note pitches may be altered by one or more semitones with *accidentals*, of which the common ones are the sharp $\sharp$ and the flat $\flat$. The sharp raises and the flat lowers note pitches by one semitone.

A series of distinct notes defines a *scale*. The most commonly used scales are the *major* scale and a natural *minor* scale which consist of seven notes. Conventionally, the notes in scales are presented in an ascending order, and the first note of the scale is called a *tonic* note which names the scale. Particularly in this work, the interval between a tonic note and another note is called a *tonic distance*. As an example of scales named by tonic notes, the C major and the natural A minor scales are represented in Figure 2.3, and the first notes of the scales are C and A, respectively. In addition, these scales contain the same seven notes, and therefore, the scales are considered to be *relative* to each other. The seven-note major or minor scale, which is used in a tune, defines the *key* of the tune. In music notation, keys are denoted by *key signatures*. If major and minor scales are relative, also the keys defined by those scales are relative, thus composing a *relative-key pair*. For example, C major and A minor keys form a relative-key pair. Specifically, relative keys are denoted by the same key signature.

The relationship between the key, chords, and melody determines the *tonality* of a musical composition. An important part of tonality is *harmony* which is implied by the succession of chords according to the rules of musical progression and modulation. Furthermore, tonality and harmony form the basis of *musical context*.

Temporal structures in music

In addition to the tonal dimensions of music, music is organised in time to construct temporal structures which are relative to the *tempo* of a musical piece. Tempo defines

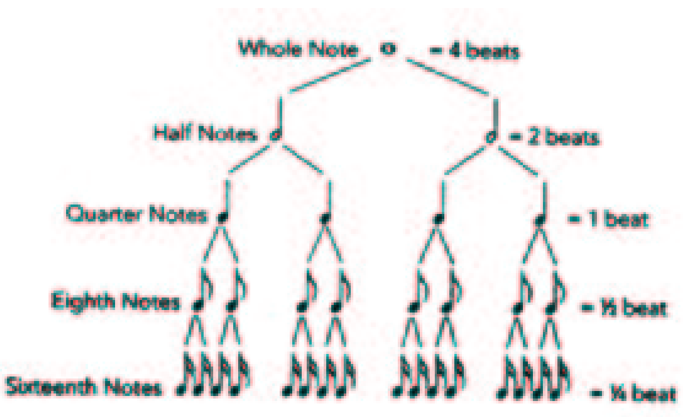

Figure 2.4: The hierarchical structure of note durations. Reprinted from `http://www.dreach.net/mus2007/theory/timeSignatures.htm`

the rate of playing the piece, and it is usually indicated by the timing of the *beat*. Beat can be considered as the most typical way to tap along music, and the absolute timing of all the temporal structures are somehow related to the beat. For example, beat usually indicates the duration of a quarter note. If the duration of a quarter note is then halved or doubled, we have an eight note or a half note, respectively. This type of construction of note durations leads to a hierarchical structure of note durations and, furthermore, temporal segments covering the musical piece. As an example, Figure 2.4 shows the hierarchical structure of note durations, and their relationship with the beat. Similarly to the note durations, *rests* (i.e., nothing is played at that time) in music have a hierarchical structure.

The temporal structures, which are larger than the duration of notes, usually divide music into equal periods called *measures*. The length of measures are indicated by the multiples of note durations, and they are indicated with *time signatures*. For example, the time signature 3/4 indicates that the measure is three quarter notes long.

2.3 Automatic melody transcription

Computational melody-transcription systems attempt to extract a meaningful sequence of notes from an acoustic input. According to our definition of melody, *pitch estimation* is an essential step in melody transcription. Nonetheless, pitch estimators (or pitch trackers) are typically only front-ends for transcription systems. In addition, the actual transcription process may include other processes, such as acoustic modelling of sound sources, or pitch analysis based on musical knowledge.

This section is organised as follows. Section 2.3.1 focuses on conventional pitch tracking algorithms that are capable of extracting a single pitch at a time. Section 2.3.2 concentrates on more sophisticated methods of multipitch estimation, and in Section 2.3.3, methods for enhancing transcription results are discussed.

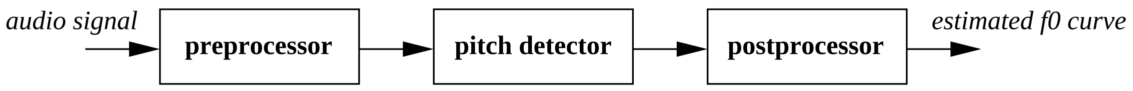

Figure 2.5: Basic steps of fundamental frequency estimation.

2.3.1 Pitch tracking algorithms

The process of fundamental frequency estimation consists of *preprocessing*, *pitch tracking*, and *postprocessing*. Figure 2.5 represents a flowchart of this process. First, the preprocessor treats acoustic signals so that the pitch detection becomes easier to perform. Common preprocessing methods include the suppression of noise and the accentuation of those acoustic features that are relevant in pitch estimation. The pitch detector extracts the pitch contour from the preprocessed signal, and finally, the postprocessor improves the pitch estimation result, for example, by correcting coarse errors and smoothing the pitch contour.

In practise, pitch is estimated in finite-length *frames* of a discrete signal. If the frame contains silence or noise, no identifiable pitch exists. Therefore, it is important to estimate also the periodicity of the signal in the frame. This is called *voicing analysis*, and there usually exists a criterion by which the voicing of the frame can be determined. Voicing analysis is typically performed by pitch estimation algorithms.

Pitch estimation algorithms may be classified in different ways. First, algorithms may be grouped to *time* or *frequency* domain algorithms according to their operating domain [Rabiner76, p. 400]. Some algorithms can be presented in both domains, and there exists also *hybrid* algorithms exploiting both processing domains. On the other hand, algorithms may be divided into *spectral place* and *spectral interval* algorithms according to those spectral properties which lead to a conclusion of the fundamental frequency [Klapuri00]. The spectral-place type algorithms handle spectral components according to their spectral location, whereas the spectral-interval type methods examine the intervals between peaks in the spectrum.

Time-domain algorithms

Autocorrelation function (ACF) is one of the most often used tools in pitch estimation. From a statistical point of view, autocorrelation is the expected value of the product of a signal with the time-shifted version of itself. Given a discrete time-domain signal $s(k)$ and a signal frame of length N, the short-time autocorrelation $r(n)$ is defined as

$$r(n) = \frac{1}{N} \sum_{k=0}^{N-n-1} s(k)s(k+n), \qquad 0 \leq n \leq N-1, \tag{2.2}$$

where n is called the *lag* and corresponds to the time shift of the signal. The short-time autocorrelation function shrinks the summation area of samples, thus decreasing the $r(n)$ values as the lag value increases. The frame length N should be

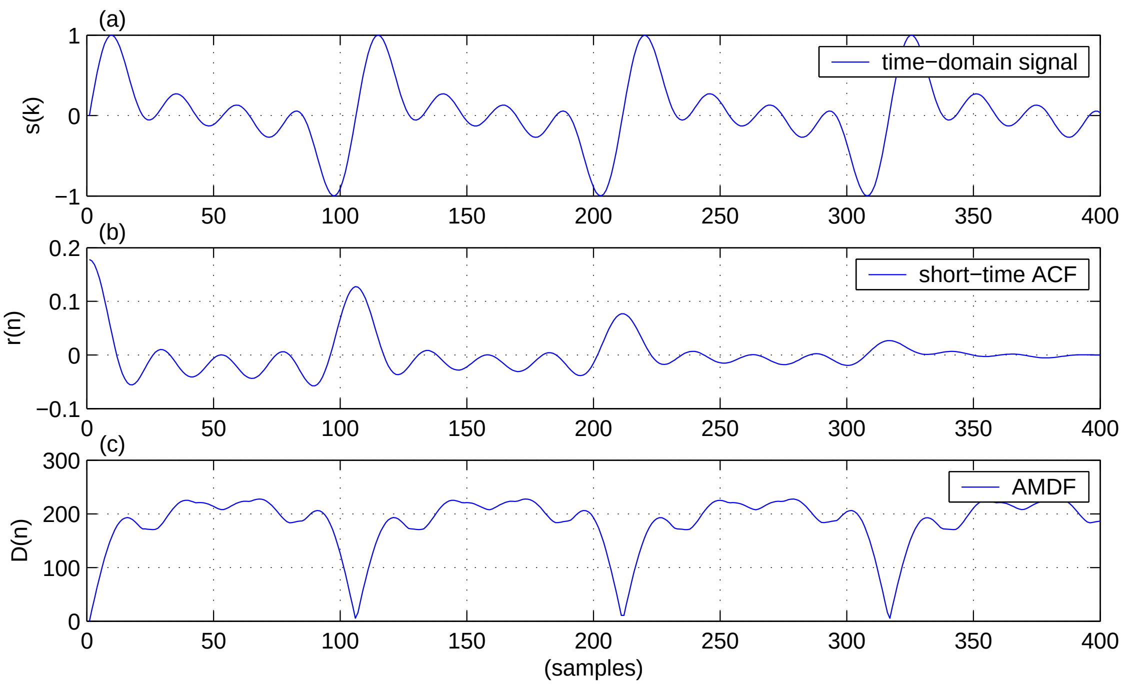

Figure 2.6: (a) Input signal (sampling rate $f_s = 21050$ Hz, $F_0 = 200$ Hz), (b) output of short-time ACF, and (c) AMDF. Peaks in (b) and (c) at $n = 105$ correspond to a fundamental frequency estimate $\frac{f_s}{n} \approx 200$ Hz.

as small as possible to preserve the time resolution, but large enough to capture at least two periods of the fundamental frequency.

The maxima in the autocorrelation function indicate the lags that correspond to the multiples of a fundamental period. Usually there are several peaks with approximately equal maxima, thus complicating the conclusion of the correct peak. However, the short-time autocorrelation prevents the detection of peaks with too-high lag values, since the values of ACF decrease as a function of n. In addition, the signal is typically preprocessed to ease pitch detection with the ACF. The autocorrelation-based pitch estimators are rather robust for noisy signals but sensitive to the spectral properties of the target sounds [Gómez03]. The short-time ACF may be classified as a spectral-place type pitch estimation algorithm, since it weights spectral components according to their locations in the spectrum [Klapuri00].

The *average magnitude-difference function* (AMDF) $D(n)$ for a time-domain signal $s(k)$ with a frame length N is defined as

$$D(n) = \sum_{k=0}^{N-1} |s(k) - s(k+n)|, \qquad 0 \leq n \leq N-1. \tag{2.3}$$

The AMDF resembles the ACF since the signal is compared with itself to detect the period. Figure 2.6 shows both the ACF and AMDF of a time-domain signal, and it shows that with the AMDF, the comparison is just based on the absolute difference between the signal frame and its time-shifted version. The more periodic the signal is, the closer to zero the minimums of the AMDF get. In practise, the level of periodicity is determined by choosing a threshold value below which the minimums of AMDF must be to satisfy a desired level of periodicity. Specifically, the AMDF is computationally lighter than the ACF, and therefore, it is preferred in real-time applications. Systems with fixed-point arithmetic, especially, benefit of the absence of multiplication.

The *cross-correlation function* (CCF) is otherwise similar to the ACF, but the analysis frame is compared to an arbitrary frame of the signal. The CCF $\chi(n)$ of a time-domain signal $s(k)$ is defined as

$$\chi(n) = \sum_{k=0}^{N-1} s(k)s(k+n), \tag{2.4}$$

where n is the lag. If $0 \leq n \leq N-1$, Equation 2.4 becomes the definition of autocorrelation. However, this is *not* required with CCF, and n can be greater than N when the frames do not overlap at all. Since the required length of the signal frame is quite big in the autocorrelation methods, the use of cross-correlation provides a better time resolution, even for low pitch estimates [Gómez03].

The *normalised cross-correlation function* (NCCF) normalises the CCF values with the energies of the compared frames. The NCCF is here defined as

$$\phi(n) = \frac{\chi(n)}{\sqrt{e_0\, e_n}}, \tag{2.5}$$

where e_l is defined as

$$e_l = \sum_{k=0}^{N-1} s(k+l)^2. \tag{2.6}$$

The normalisation reduces the influence of rapid changes in the signal amplitude, and the fundamental period (and its multiples) can be found at the lag values n for which $\phi(n)$ is close to 1. The NCCF is robust for noisy signals, and it was successfully applied in Talkin's RAPT pitch-tracking algorithm [Talkin95], for example.

The *envelope periodicity* (EP) pitch-estimation models are based on the observation that there exists periodic fluctuation in the amplitude envelope of complex tones. If a tone consists of several harmonics with an approximately same frequency interval between each other, they will produce beating in the amplitude envelope. Consequently, the time between the envelope peaks corresponds to the fundamental period. Since the frequency distance between harmonics is detected, EP models are classified as spectral-interval type algorithms.

The envelope periodicity pitch-tracking procedure is the following. A time-domain signal is first half-wave rectified and then lowpass filtered to reveal the shape of the amplitude envelope of the signal. The periodicity of the amplitude envelope is finally detected, for example, with the autocorrelation function. The EP pitch-tracking stages are illustrated in Figure 2.7.

To approach the pitch detection problem from a more psychoacoustic viewpoint, the input signal may be divided into frequency channels to mimic the function of human

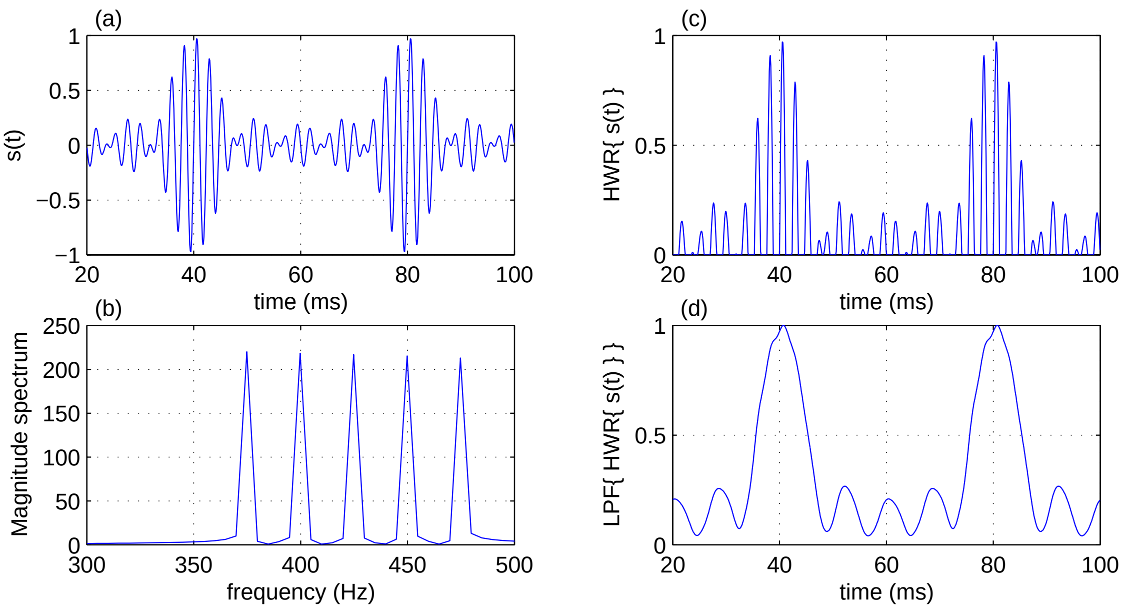

Figure 2.7: (a) A time-domain signal $s(t)$ including only a few multiples of $F_0 = 25$ Hz; (b) the magnitude spectrum showing the signal harmonics; (c) the half-wave rectified signal HWR$\{s(t)\}$; and (d) the amplitude envelope LPF$\{$HWR$\{s(t)|\}\}$ of the signal. The peak period in the amplitude envelope $t_\Delta = 40$ ms reveals the fundamental frequency estimate $\frac{1}{t_\Delta} \approx 25$ Hz.

auditory system. EP modelling is performed for each channel separately, and finally, the results are combined to obtain a fundamental-frequency estimate. In this case, the method is spectral-interval and spectral-place algorithm at the same time. One important example of such a model is the *unitary pitch model* proposed by Meddis and O'Mard in [Meddis97].

Frequency-domain algorithms

Frequency-domain pitch estimation algorithms process the signal in frequency-domain at some stage of the pitch estimation. This obviously requires transforming time-domain signals into the frequency domain, e.g., with the discrete Fourier transform (DFT).

The *cepstrum* $c(n)$ of the signal $s(k)$ is the inverse Fourier transform (IDFT) taken from the logarithm of the magnitude spectrum [Gómez03]:

$$c(n) = \mathrm{IDFT}\{\log|\,\mathrm{DFT}\{s(k)\}|\}. \tag{2.7}$$

Since the ACF can also be calculated in the frequency domain as presented in Equation 2.8, the cepstrum analysis and the ACF methods are quite similar; cepstrum only uses the logarithm instead of the second power.

$$r(n) = \mathrm{IDFT}\{|\,\mathrm{DFT}\{s(k)\}|^2\} \tag{2.8}$$

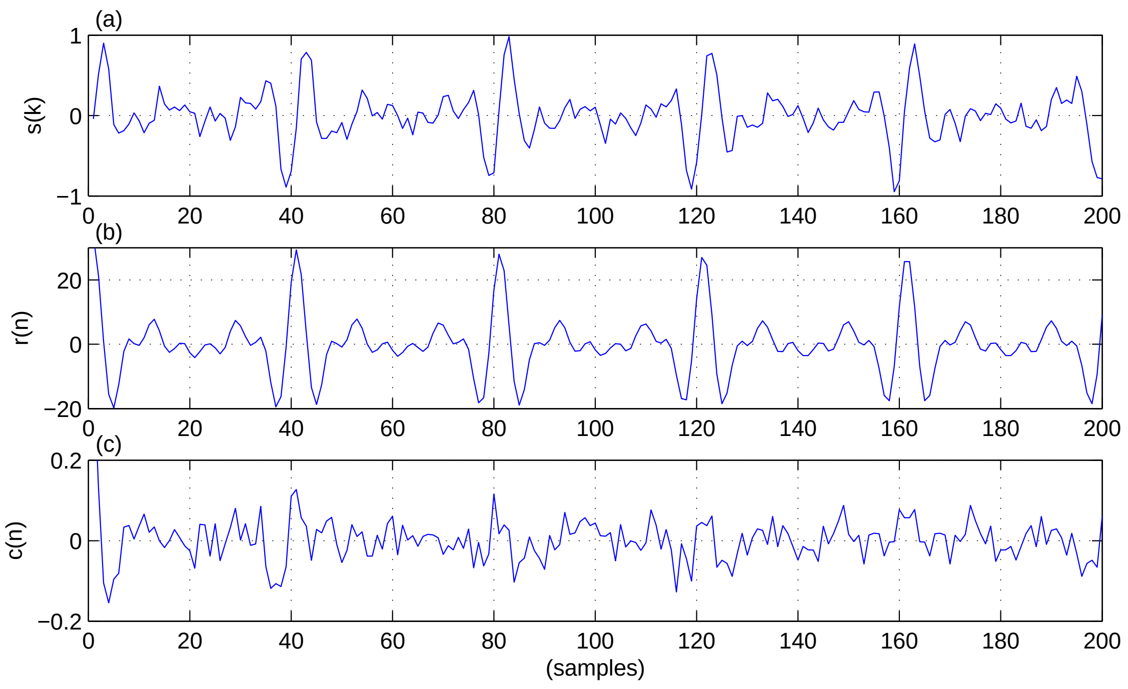

Figure 2.8: (a) A noisy time-domain signal $s(t)$; (b) the ACF calculated with Equation 2.8; and (c) the cepstrum calculated with Equation 2.7.

However, there exists a significant difference in noise robustness to the ACF's credit, despite the analogy of the compared methods. This is illustrated in Figure 2.8 in which the ACF and the cepstrum are evaluated for a noisy signal. On the other hand, the cepstrum-based pitch estimation performs well for speech sounds with strong formant structures, and therefore, it is widely used in speech-analysis systems. Similarly to the autocorrelation methods, the cepstrum method is classified as a spectral-place type pitch estimation algorithm.

The *spectral autocorrelation* pitch estimation method employs the properties of the spectrum of harmonic sounds. As presented in Section 2.1.1, the harmonic partials of a complex tone are separated by an equal frequency interval from each other, and the most evident frequency interval, which corresponds to the fundamental frequency, is detected by calculating the ACF of the magnitude spectrum. If a complex tone has stronger harmonics, e.g., on every second multiple of the fundamental frequency, twice too low pitch estimates are likely produced, whereas twice too high estimates are improbable. The spectral autocorrelation method is classified as a spectral-interval type method.

Harmonic matching methods search for peaks in the magnitude spectrum, measure the harmonic relations of these peaks, and collect them to determine the fundamental frequency. One such method was proposed in [Piszczalski79], in which after the spectrum peak detection, pairs of these frequency components are constructed. If the frequency ratio of a component pair is close enough to a small integer ratio, the pair is considered as an ideal harmonic pair. The ideal harmonic pairs are tabulated and weighted according to the peak amplitude values, particularly favouring the harmonic pairs with small integer ratios. Finally, the decision of the fundamental frequency is made according to the most promising harmonic relation.

2.3.2 Multipitch estimation

Multipitch estimation tries to extract multiple simultaneous pitches in mixture signals. The approach to multipitch estimation is somewhat different than to single-pitch estimation; instead of pinpointing a single pitch from signals, a more comprehensive view of the *auditory scene* is needed, since there is a mixture of sounds involved. As a consequence, there are several approaches towards solving multipitch estimation problem: perception of sound, physiology of hearing, rules governing musical sources, and physics of the modelled sound source [Klapuri03a]. In addition, multipitch analysis often utilises the research made on the *computational auditory scene analysis* (CASA), and sound source separation within psychoacoustics. Multipitch algorithms are most often developed for music application purposes which facilitate a more discriminating sound analysis with the aid of knowledge on music theory and on instrument sounds. On the other hand, the wide variety of musical signals *is* the problem. The complexity and the performance of multipitch-estimation systems depend on the input-sound attributes, including the number of allowed simultaneous sounds and instruments, pitch range, and the interference of noise and irrelevant sounds.

The unitary pitch model [Meddis97], mentioned in Section 2.3.1, modelled the physiology of the human auditory system by dividing the audible frequency range into subbands. Although the unitary model performed no detection of multiple pitches, the principle of subband independent pitch analysis was the motivation for an *auditory-model based* approach to the multipitch estimation problem [Klapuri03a]. The unitary model principles were applied to multipitch analysis by Cheveigné and Kawahara in [Cheveigné99], where they introduced another significant idea based on psychoacoustics, the iterative cancellation of detected sounds. This means that once a sound is detected in the input signal, it is removed and the residual is reprocessed. However, the algorithm is computationally exhaustive due to the multichannel auditory filterbank.

A computationally more efficient multipitch estimation algorithm was proposed in [Tolonen00]. The core of the idea is to (i) separate the signal into two frequency channels of high and low frequencies; (ii) perform half-wave rectification and low-pass filtering for the high frequency channel; (iii) detect the periodicities with the autocorrelation function for both channels; and (iv) sum the channels to gain the summary autocorrelation function (SACF). In addition, the model includes preprocessing of the input signal and postprocessing of the SACF to produce an enhanced SACF. The enhanced SACF is used to determine the fundamental frequencies, thus making the iterative sound cancellation unnecessary.

In addition to the auditory-based approach, a *perceptual* approach to analyse multipitch sensation is closely related to the auditory scene analysis. The principles of sound perception are employed to determine what elements of sound are meaningful and how they are processed by humans. For example, Kashino *et al.* presented a hierarchical structure of acoustic entities (i.e., frequencies, notes, and chords) comprising musical sounds [Kashino95], and applied it in an automatic transcription system for polyphonic sounds. In addition to the knowledge on auditory event perception, high-level musical information on chords, instrument timbre models, and tone memories were utilised in a probabilistic manner.

Considering the multipitch estimation problem purely from a physical point of view, *signal-model* based systems have been developed in which the aim is to find the physical parameters of observed signals. One such signal model by Davy and Godsill [Davy03] is defined as

$$y(t) = \sum_{i=1}^{K} \{a_i(t)\,\omega_i \cos[(i+\delta_i)\omega_0 t] + b_i(t)\,\omega_i \sin[(i+\delta_i)\omega_0 t]\} + v(t), \tag{2.9}$$

where t denotes time. The unknown parameters are the fundamental frequency ω_0, the number of frequency partials K, the partial amplitudes $a_i(t)$ and $b_i(t)$, and the detuning parameter δ_i. The term $v(t)$ models the noise component. As a result, the model considers time-varying amplitudes, detuned harmonics, and the presence of noise. However, the parameter space is enormous even for a single fundamental frequency, despite the assumptions made about musical signals. Authors reported no extensive simulation results, but a mention that the system was able to simultaneously detect up to three fundamental frequencies.

2.3.3 Melody transcription systems

Automatic melody transcription involves two main problems: tracking pitch sequences in music and converting these pitch sequences into a symbolic notation. The former problem has several algorithmic solutions as discussed earlier. The latter problem, however, is much more difficult since musical performances are seldom perfect. For instance, if a performer sings a wrong note, the audience almost certainly notices the false note in the performed melody. If this same performance is analysed by computers, the false note sounds as good as any other note, since computers as such do not “understand” music. In the following, approaches to solve this problem are discussed.

Acoustic modelling refers to the assumptions concerning the acoustic properties of input signals. The model may be based on simple presumptions of the input signal like the pitch range of a certain instrument, or, a more intelligent model may consider the temporal behaviour of note pitches, for instance. *Musicological models*, on the other hand, apply knowledge of high-level musical structures during the interpretation of melody contours. For example, some notes or intervals are more frequent than others in a given musical context.

Several melody transcription systems make assumptions about acoustic musical sounds. These acoustical features are commonly employed already at the pitch detection stage, for example by stressing the frequency range of fundamental frequencies. Nonetheless, overly restricting conjectures about acoustic signals limit the system robustness and flexibility. Statistical acoustic models have been successfully used in speech recognition, whereas the potential of modelling acoustic properties of sound events is not much exploited in melody transcription systems. Furthermore, models considering the temporal evolution of sound events are quite uncommon. Shih *et al.* proposed one such acoustic model for humming sounds in [Shih03a]. Each note was modelled with a three-state left-to-right hidden Markov model, and also the note duration was modelled.

Musicological models are more commonly used in automatic music transcription systems than acoustic models, since acoustic modelling requires a lot of effort, including an acoustic database and the model training. Musicological models may be based either on musical rules or on estimated attributes of musical performances. Moreover, a probabilistic approach to musicological models enables the evaluation of several competing interpretations [Klapuri03a]. In general, musicological modelling is not restricted to melodies or harmony, but all musical structures may be exploited. For example, the music scene analysis system proposed by Kashino *et al.* [Kashino95] uses three levels of abstraction: frequency components, musical notes, and chords. In addition, notes and chords are analysed to take the temporal continuity of these abstractions into account. Another system by Martin [Martin96] applied the knowledge on musical practise of notes, intervals, and chords to transcribe four-voice Bach chorales. The main issue in his article was to propose a computational framework for a music analysis system. However, the implemented system was unsuitable for other kinds of music because of the coarse classification of musical events. One of the state-of-the-art systems by Viitaniemi *et al.* modelled the note transitions and durations of sung notes with a hidden Markov model [Viitaniemi03]. In addition, a key signature estimation method was proposed to emphasise the predominant note occurrences in the musical context.

2.4 Automatic rhythm transcription

There are two main problems in automatic rhythm transcription: beat-tracking (or, tactus-tracking) and the transcription of rhythm sequences. The term beat-tracking refers to the estimation of the underlying meter in music signals, and the rhythm-sequence transcription aims at transcribing individual sound events in rhythm sequences. Rhythm transcription may be performed for three types of music signals: purely percussive signals, percussive signals including other instruments, and non-percussive signals. Both beat-tracking and rhythm-sequence transcription can be performed for all these signal types, nevertheless, music signals with percussive sounds are more intuitive inputs for rhythm-sequence transcription than non-percussive signals.

Automatic rhythm transcription has gained relatively little attention compared to melody transcription. In addition, the research has focused more on classification of isolated percussive sounds than on rhythmic sequence transcription. However, due to the importance of drums and percussive instruments in contemporary music, rhythm sequence transcription systems with many possible applications have become more interesting.

Automatic rhythm transcription is discussed here to inspire the integration of melody and rhythm transcription systems. In the following, sections 2.4.1 and 2.4.2 discuss about beat-tracking and rhythm-sequence transcription systems, respectively.

2.4.1 Beat-tracking

Beat-tracking aims at estimating the tactus of music, but only few systems have been proposed that are capable of doing this extraction from realistic audio signals. The main difficulty is that the beat structure is not necessarily explicitly expressed in music, but it is implied by the relations of various musical elements. In addition, the metrical complexity of different audio signals affects the performance of beat-tracking systems.

Extraction of temporally essential musical elements from acoustic signals relies on the detection of the *phenomenal accents* which are the moments in music having perceptual emphasis [Lerdahl83]. For example, changes in chords, notes, and in timbral intensity may be considered as phenomenal accents.

Goto *et al.* proposed a real-time beat-tracking system in [Goto98] which used detection of note onset times, chord changes, and drum patterns in real-world music signals. Their beat structure recognition consists of detecting the beat-tracking cues in music signals, interpretation of those cues to find out the beat structure, and managing the ambiguity of the interpretation. The proposed system was evaluated with their own quantitative measure system on a smallish amount of popular music audio excerpts, and the system could rather well recognise quarter notes, half notes, and measure levels from commercial audio excerpts with or without drums. However, the time signature was assumed to be 4/4, and the applicable range of tempos was somewhat limited.

Another beat-tracking system proposed by Scheirer in [Scheirer98] analyses tempo and beat from arbitrary music signals. The system uses a filterbank to divide the input signal into subbands, and the amplitude envelope and its derivative are calculated at each subband. Another filterbank of tuned resonators phase-lock to the frequencies of periodic modulation of the envelope derivatives, and the subband results are summed up to arrive at the tempo estimate for the signal. Once the tempo is estimated, the phase of the signal is determined by examining the resonator filter state vectors. The described system was implemented in C++ and evaluated in both qualitative and quantitative manners. System performance was tested with 60 music excerpts from different musical genres, and some listening tests were carried out to validate the system's human-like beat-tracking.

Klapuri proposed a method for musical meter estimation at three metrical levels (tatum, tactus, and measure) for acoustic music inputs [Klapuri03b]. This method is applied in our transcription system and described in Section 3.2.2. As a mention, the system was evaluated with a broad database of annotated music recordings from different genres. In addition, a comparison to earlier methods, including [Scheirer98],

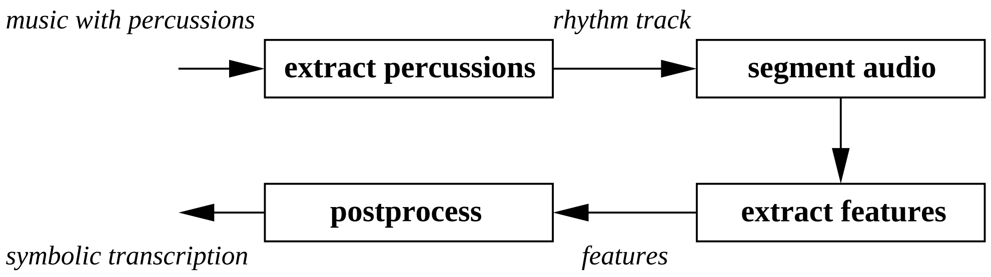

Figure 2.9: Basic steps of rhythm sequence transcription.

was accomplished. The system turned out to be rather robust tatum and tactus tracker, whereas the measure-level estimation performed less successfully.

2.4.2 Rhythm sequence transcription

Rhythm sequence transcription aims at extracting rhythm sequences from music and producing labels for the sound events that occur in the sequences. Figure 2.9 shows a common procedure of the rhythm sequence transcription. The first task is to extract the percussive sounds or the so-called *rhythm track* from a music signal. The rhythm track is segmented into temporal blocks, usually based on meter estimation and beat-tracking information. Then, the segmented blocks are processed to extract features that are used to determine the percussive sounds in the signal. Finally, the extracted features are postprocessed to produce a symbolic transcription of the rhythm sequence.

Reasonable rhythm tracks are considered to contain at least a few different percussive sounds which need to be separated and recognised. This *labelling* problem is conventionally approached from two perspectives [Gouyon01]. First, the perceptual classification attempts to mimic human perception of sound signals, and it requires no previous knowledge of the sounds to be separated. On the contrary, the second approach is to provide training material for the percussive sounds to be classified, and conventionally, acoustic models of the percussive sounds are applied. In general, the use of acoustic models limits the type of input signals but provides a better transcription performance. The labelling problem is considered, for example, in [Gouyon01] in which different labelling techniques are compared.

Paulus and Klapuri proposed a system for transcribing polyphonic drum sequences from acoustic signals [Paulus03b]. The system uses meter estimation to establish a time grid to which possible drum events are associated. Drum events are labelled with an acoustic model, which was built from a drum-sound database, to determine whether there occur drum events or silence at certain points on the time grid. Finally, the rhythmic periodicity between the drum events are examined with a statistical N-gram prediction model to produce a transcribed rhythm sequence. In particular, the use of N-gram prediction improves transcription results. The system does not separate different drum events from each other but directly recognises the combinations of these.

Later, Paulus and Klapuri proposed a system capable of transcribing percussive audio signals with three arbitrarily chosen percussive sounds [Paulus03a]. The aim of the system was to recognise the percussive sound events and to give each event a rhythmic role label. At first, the meter estimation and the note-onset detection is performed. Then the features are extracted only at the instants of the detected onsets and clustered with the fuzzy K-means algorithm. Finally, each cluster is labelled with the rhythmic roles by using meter estimation information and a probabilistic model. The probabilistic model gives likelihoods for each label to occur at different metrical position, for which the likelihoods were estimated from a database of rhythm sequences. In the simulations of the system, non-standard percussive sound sets were created (including foot-tapping, pencil-clicking, and percussive vocal sounds) to synthesise audio with arbitrary sound sets from the database. The evaluation of the system was accomplished in several parts, and the results were promising. It was also found that some music genres were easier to transcribe than others.

2.5 Applications of music content analysis

Music content analysis enables applications that process musical input, turn it into meaningful musical information, and use it in a desired way. If we are able to automatically analyse musical information, the enabled applications are not only restricted to analysis applications but also modelling, synthesis and interaction with music signals becomes possible, thus providing an unbounded application area. This section reviews a few applications that are enabled by music content analysis, excluding the automatic transcription of music: automatic music retrieval, object-based coding of music, and musical interaction with computers. In addition, the research on music content analysis includes, for example, musical genre classification and instrument recognition. They are not discussed in this thesis, but an interested reader is referred to [Heittola03].

2.5.1 Music retrieval

Since the number of digital-music databases is rapidly increasing, researchers have developed methods for automatic music retrieval from music databases. Music in the databases is stored in such a format that only the relevant parts of music, such as melody contour, lyrics, and accompaniment are preserved. This is essential to enable the identification and the indexing of the music excerpts. If we wish to search a music excerpt from the database, we need to perform a musical query that identifies the excerpt. Musical queries are mainly based on melodies, because for common users, melodies are the most intuitive way for distinguishing music excerpts from each other. However, the variety of different musical representation techniques complicates unambiguous indexing of music excerpts.

The preferred strategy for making musical queries has been queries by *humming*, although more exotic user interfaces have also been proposed. The query-by-humming

(QBH) systems use conventional speech and audio pitch-analysis techniques as retrieval-system front-ends and apply various error-correction techniques to produce more accurate melody queries. Reviews of QBH systems are presented, for example, in [Pollastri02, Shih03b].

2.5.2 Object-based coding of music

Music signals contain lots of redundancy both in low-level details of signal waveforms and in high-level musical structures and performance techniques. The object-based coding of music tries to reduce this redundancy by representing music as *objects* at different abstraction levels in music signals, striving for the compression of musical information. The more different levels of abstraction are exploited, the more elimination of redundancy is achieved. *Perceptual* compression methods remove the redundant information from sound signals that is irrelevant to human listeners. On the other hand, model-based compression techniques decrease the *process redundancy* of sounds, i.e., the redundancy in the physical creation of certain sounds. In practise, a sound source is approximated with a parametric model which mimics the physical production of the sound. The parameters of the model are estimated from a sound signal, and the original signal may be later re-synthesised from this parametric representation.

As an example of the object-based coding of music, the idea of the *structured audio* (SA) coding is to establish a formal language for describing musical information. The SA decoder was developed to enable the transmission of synthetic sounds and music with a very low bandwidth, and it is a part of the MPEG-4 Standard [MPE98]. The *Musical Instrument Digital Interface* (MIDI) protocol, a coarse predecessor of SA, stores the attributes of a music performance such as notes, tempo and instruments. Nevertheless, MIDI does not specify what the musical performance should *sound* like, although there are sound fonts that can be used as instruments. SA solves this problem with a hybrid of the musical information representation and the description of sound synthesis. In contrast to conventional audio codecs, which code acoustic signals with a fixed model, SA provides a description language for sound-source models. As a result, the original music performance may be delicately restored in the decoder.

As an important part of the SA, the *structured audio orchestra language* (SAOL) specifies the methods for creating instrument sounds. It is not limited to any particular sound synthesis method, but it is a general description language for sound synthesis. A SAOL synthesiser is controlled with the *structural audio score language* (SASL) which is a simple representation of music resembling to MIDI. Despite the fact that SA is a part of the MPEG-4 standard and provides the specification of the decoder, the lack of proper SA music encoders makes SA yet a relatively untapped format. Computer music systems that are capable of adequate music and sound analysis are under development, and in the future, they will enable an extensive use of SA.

2.5.3 Interactive music systems

Perhaps the most challenging task for computational music-analysis systems is the competence for interaction with music signals. Besides the ability to receive and understand music, the computer is expected to produce a rational response to its input, even in real-time. Computational music cognition is a composite of knowledge in music theory, cognitive science, and artificial intelligence [Rowe01]. Rules concerning musical structures combined to their human perception [Lerdahl83, Temperley01] set up the groundwork for this musical process. Artificial intelligence emulates the human-like reasoning to produce a sensible reaction to the observed events in music.

A typical interactive application would be an accompaniment system which adapts to an ongoing music performance and produces a real-time musical response to it. An interactive improvisation system goes even further. At present, realistic implementations of interactive music systems are often restricted to a certain music style and have a limited number of observed input sources. Rowe examined these ideas and provided some MIDI software solutions in his book “Machine Musicianship” [Rowe01].

In addition, musical information may be used as an input for interactive multimedia environments. In such applications, computational interpretation of music controls various multimedia resources, such as live performance stage-lightning and computer graphics. For example, Goto *et al.* proposed an application where an audio-based beat-tracking system guides computer visuals in real-time [Goto98].

3 Musical Feature Extraction

Musical information consists of *tonal* and *rhythmic* components, roughly corresponding to the frequency dimension and the time dimension of audio signals, respectively. These two concepts constitute the background for music-signal analysis and for the extraction of *musical features* from audio signals. Musical features, such as pitch, voicing, and phenomenal accents, reflect the musical content of music signals. Musical features are interpretations of *acoustic features*, such as the frequency content, silence, and noise. Since we interpret acoustic features in a musical sense, it is more appropriate to discuss about musical feature extraction than acoustic feature extraction. In the same way, humans interpret acoustic features to derive musical sensations.

The automatic transcription of melodies may exploit both the tonal and the rhythmic components of music. The tonal component contains note pitches, enabling the analysis of notes in general. The rhythmic component, on the other hand, enables the analysis of *note sequences* since the starting points and the durations of notes are also an important part of melodies. The desired musical features can be extracted from audio signals with various audio processing algorithms. These algorithms act as a *front-end* for the analysis system, and their accuracy has a significant influence on transcription results.

This chapter represents the algorithms used to extract both the tonal components and the rhythmic components from audio signals in the proposed transcription system. Section 3.1 discusses *pitch extraction* (i.e., the extraction of the tonal component), and Section 3.2 explains the algorithms and methods used to extract the rhythmic components of melodies.

3.1 Pitch extraction

Pitch extraction is the most fundamental step in melody transcription. Throughout this thesis, pitch estimates are represented as *MIDI-note numbers* to clarify the relationship between pitch-estimate frequencies and the corresponding musical notes. This relationship is expressed by Equation 3.1,

$$x' = 69 + 12\log_2\left(\frac{F_0}{440}\right), \tag{3.1}$$

where F_0 is an estimated pitch frequency and x' is the corresponding MIDI-note number. The numbers 69 and 440 represent a MIDI-note reference, since the MIDI-note number 69 conventionally has a fundamental frequency 440 Hz. Moreover, number 12 indicates that an octave is subdivided into twelve notes.

The proposed transcription system uses a pitch-extraction algorithm represented in Section 3.1.1. In addition, pitch estimates are postprocessed by *tuning* for which an algorithm is proposed in Section 3.1.2.

3.1.1 YIN algorithm

Several solutions have been suggested to estimate the pitch of an audio signal, including the *YIN algorithm* proposed by de Cheveigné and Kawahara [Cheveigné02]. The YIN algorithm is based on average magnitude-difference function and extended with several modifications to improve the pitch estimation accuracy. Although the algorithm is thoroughly presented and evaluated by the original authors, the algorithm and its parameters are briefly explained in the following. In addition, the algorithm steps are clarified by an example.

Given that s is a discrete time-domain signal, N is the frame length, and κ is a constant threshold value, the YIN algorithm is formulated as follows:

1. Calculate the squared difference function $d(\tau)$ for a desired range of lag values τ:
$$d(\tau) = \sum_{n=0}^{N-1} (s(n) - s(n+\tau))^2 \tag{3.2}$$

2. Evaluate the cumulative mean normalised difference function $d'(\tau)$:
$$d'(\tau) = \begin{cases} 1, & \tau = 0 \\ d(\tau) \,/\, [(1/\tau)\sum_{j=1}^{\tau} d(j)], & \text{otherwise} \end{cases} \tag{3.3}$$

3. Search the smallest value of τ for which a local minimum of $d'(\tau)$ is smaller than a given absolute threshold κ. If no such value is found, search the global minimum of $d'(\tau)$ instead. Let $\hat{\tau}$ be the found lag value.

4. Interpolate the $d'(\tau)$ function values at abscissas $\{\hat{\tau}-1, \hat{\tau}, \hat{\tau}+1\}$ with a second order polynomial. Search the minimum of the polynomial in the range $(\hat{\tau}-1, \hat{\tau}+1)$ to obtain an estimate of the fundamental period.

The YIN algorithm has several features that improve the algorithm accuracy and robustness. The second step normalises the squared difference-function values with its average over the shorter lag values, thus preserving the function $d'(\tau)$ values high for smaller lag values. This prevents the algorithm from choosing a too small lag value as the fundamental period estimate. In addition, the second step normalises the function so that an absolute threshold value κ can be applied to find the first

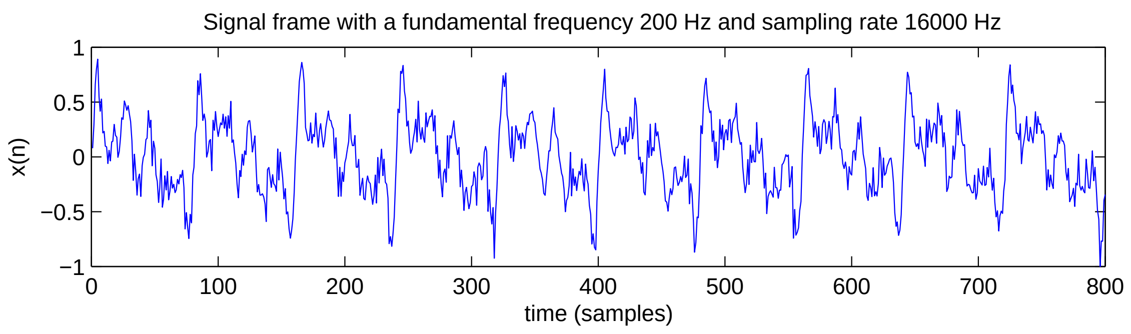

Figure 3.1: An example of an input-signal frame for the YIN algorithm.

function minimum, i.e., the input signal level does not affect the algorithm performance. The third step uses the threshold value to define the set of acceptable fundamental period values. Finally, the fourth step interpolates the $d'(\tau)$ function to enhance lag value resolution; otherwise, the fundamental period values would be restricted to integer values.

The YIN algorithm was chosen as the front-end for our transcription system since it is relatively robust and computationally light. In this work, a frame length of 25 ms, a threshold value of 0.15, and a maximum lag value of 25 ms are used. In addition, the value $\nu = d'(\hat{\tau})$ is used to indicate *voicing* (i.e., the degree of periodicity) within the signal frame; if $\nu \leq \kappa$, the frame is considered to be voiced. Consequently, the YIN algorithm extracts *two* musical features within each signal frame: a fundamental frequency estimate F_0, and a voicing value ν. Fundamental frequency values are mapped to pitch estimates x' according to Equation 3.1.

The steps of the YIN algorithm are illustrated by the following example. Figure 3.1 represents a 50 ms segment of a time-domain signal containing frequency components at the fundamental frequency 200 Hz and at its multiples, and additive noise. The signal is sampled at 16 kHz rate, and the above-described algorithm parameters are applied. Particularly, two signal frames are required to evaluate the squared difference function for the desired lag-value range. The example-signal frames are processed step-by-step with the YIN algorithm, and the intermediate results are illustrated in Figure 3.2 for each algorithm step. Panels (a) and (b) represent the squared difference function and the cumulative-mean normalised functions, respectively. Clearly, the function values in panel (b) decrease as the lag values increase, and the function hovers around unity. Subsequently, the first minimum of $d'(\tau)$ is found at lag value $\hat{\tau} = 81$ satisfying the threshold condition $d'(\hat{\tau}) < \kappa$. Finally, panel (c) shows the parabola fitting, resulting in a fundamental frequency estimate of 198 Hz, and a voicing value of 0.11. According to our voicing criterion, this frame is voiced.

The YIN algorithm is used in the transcription system to produce pitch and voicing estimates which are considered as musical features. Figure 3.3 shows these two features extracted from a melody that has been performed according to a reference which the performer heard through headphones during the performance. The notes of the accompaniment are called reference notes, and they are shown in panel (a).

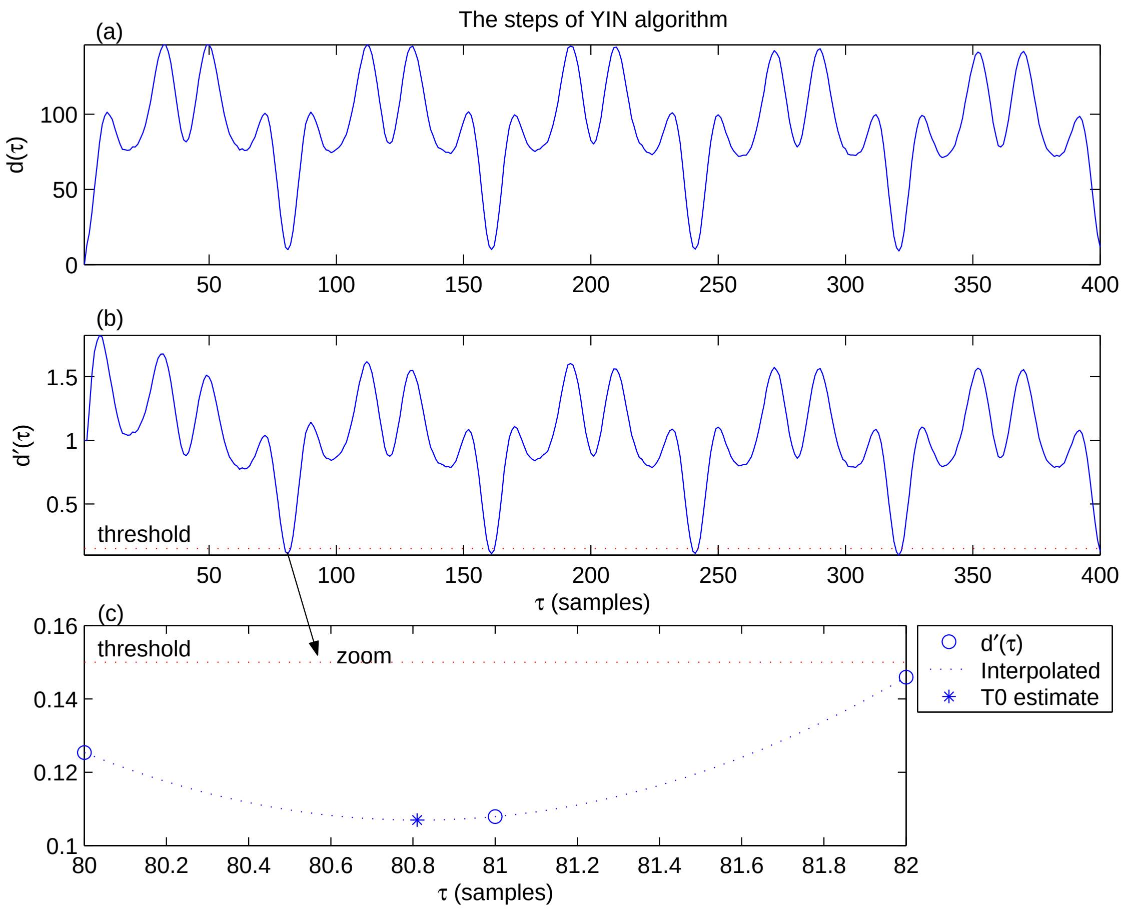

Figure 3.2: YIN algorithm steps for the input signal presented in Figure 3.1: (a) the squared difference function $d(\tau)$, (b) the cumulative-mean normalised difference function $d'(\tau)$, and (c) interpolated $d'(\tau)$ near $\hat{\tau}$. The fundamental period estimate $T_0 = 80.81$ samples corresponds to a fundamental frequency estimate $F_0 \approx 198$ Hz. The voicing value is $\nu = d'(T_0) \approx 0.11$ for the signal frame.

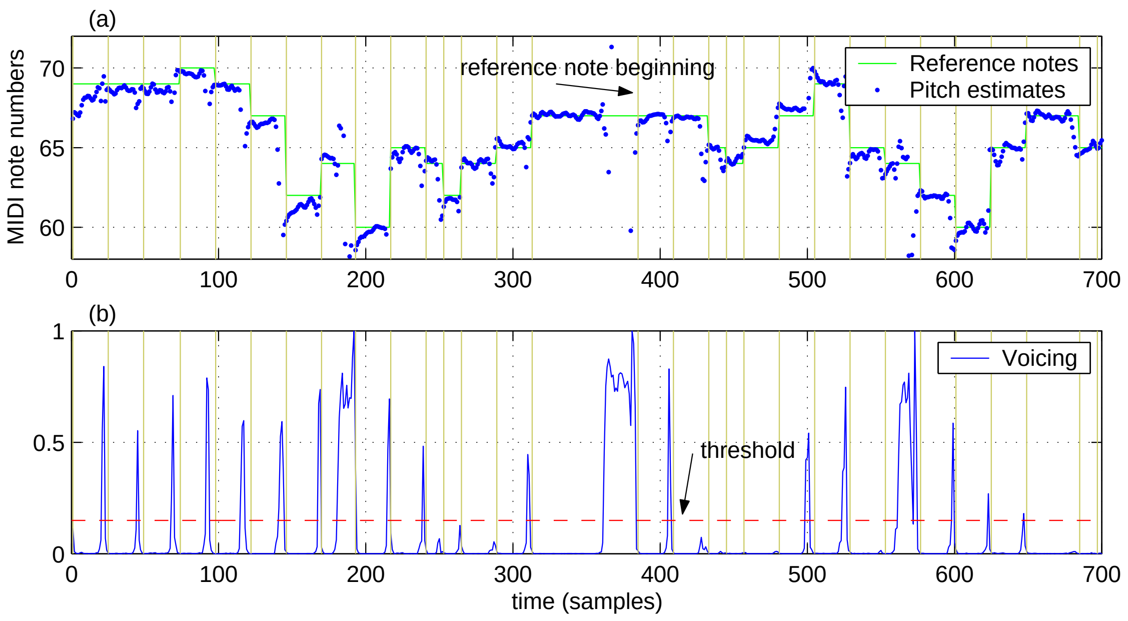

Figure 3.3: Musical features extracted by the YIN algorithm: (a) pitch estimates and (b) voicing.

In addition, the beginnings of the reference notes are denoted with vertical lines. As a result of the YIN algorithm, the pitch sequence and the voicing of the performance are illustrated in panels (a) and (b), respectively. Note that *small* voicing values indicate clear periodicity. The pitch estimates are quite accurate when the voicing $\nu \approx 0$. However in the note endings, the voicing value abruptly increases as a result of silence or noise in the performance. In those cases, the pitch estimates are not reliable and get almost arbitrary values.

3.1.2 Tuning of pitch estimates

Only few people can sing in absolute tune without any accompaniment, or a reference level of the absolute tuning. For example, a singer without any reference may start singing between two note pitches. Moreover, non-professional singers tend to change their tuning (typically downwards) during long melodies, thus complicating the transcription process. Therefore, a tuning algorithm is proposed for pitch estimates represented as MIDI-note numbers, and it is designed to compensate for the drifting in tuning. The proposed algorithm is recursive and, therefore, efficient and usable also in real-time applications.

Given a pitch estimate x'_t satisfying the voicing criterion at time t, and a *leak factor* $\alpha \in [0, 1]$, the recursive tuning algorithm is defined as follows:

1. Initialisation ($t = 0$):

- Initialise a grid of tuning values $\mathbf{g}$.

$$\mathbf{g}_0 = [-0.4, -0.3, \ldots, 0.4, 0.5]^T \in \mathbb{R}^{10\text{x}1} \tag{3.4}$$

- Initialise a tuning histogram $\mathbf{h}$ with an appropriate mass γ.

$$\mathbf{h}_0 = \gamma \mathbf{1} \in \mathbb{R}^{10\text{x}1} \tag{3.5}$$

- Initialise a histogram mass centre $c \in \mathbb{R}$.

$$c_0 = 0 \tag{3.6}$$

2. Recursion ($t > 0$):

- Search the maximum value of vector

$$\mathbf{m} = |\mathbf{g}_t + x'_t - \lfloor \mathbf{g}_t + x'_t \rfloor - 0.5| \tag{3.7}$$

  and denote its index with i. Set all the other elements of the vector to zero, i.e., $m_{k \neq i} = 0$. Add this vector to the tuning histogram weighted by the leak factor α.

$$\mathbf{h}_t = \alpha \mathbf{h}_{t-1} + (1 - \alpha)\mathbf{m} \tag{3.8}$$

  The $\lfloor \mathbf{x} \rfloor$ is the *floor* function giving the largest integers less than or equal to each element of $\mathbf{x}$. In particular, the $|\mathbf{x}|$ operation takes here the absolute values of each vector element, not the vector norm.

- Update the histogram mass centre.

$$c_t = \alpha c_{t-1} + (1 - \alpha) \left[\mathbf{h}_t^T \mathbf{g}_t / \sum \mathbf{h}_t \right] \tag{3.9}$$

- Add the histogram mass centre value to the pitch estimate x'_t to obtain the tuned pitch estimate x_t.

$$x_t = x'_t + c_t \tag{3.10}$$

- Shift the tuning grid so that the median of the grid (in this case, the fourth element of $\mathbf{g}$ vector) is always nearest to the histogram mass centre c_t. This is done by adding (or, by subtracting) a multiple of 1/10 to $\mathbf{g}$, since the tuning grid was initialised with ten elements. Moreover, each element in the tuning grid corresponds to a histogram element, so the histogram elements must be also shifted in order to calculate Equation 3.9. This is done by taking the elements from the beginning of $\mathbf{h}$ to the end of $\mathbf{h}$, or vice versa, depending on the direction of the shift. The number of the shifted elements is equal to the number of the multiples of 1/10.

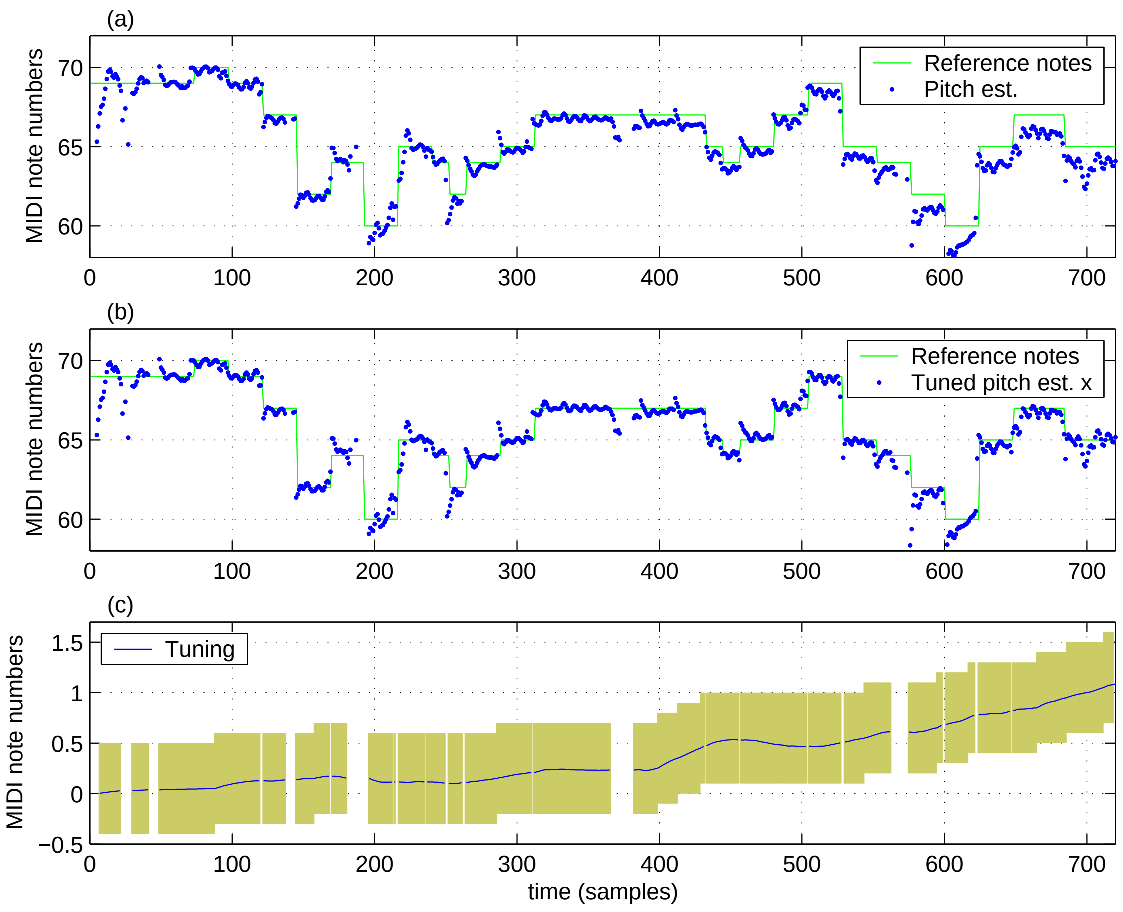

Figure 3.4: Tuning of pitch estimates: (a) measured pitch estimates x'_t with a descending trend; (b) tuned pitch estimates x_t; and (c) the histogram mass centre c_t and the tuning grid values as a shaded box.

The tuning algorithm is based on two vectors: a tuning grid $\mathbf{g}$, and a tuning histogram $\mathbf{h}$. The tuning grid defines the set of tuning values, and the tuning histogram accumulates the occurrences of points on the tuning grid that tune the pitch estimates near the absolute tuning. The mass centre of the tuning histogram c is used to tune the pitch estimates. By using the histogram mass centre, we obtain a smooth tuning process instead of using rapidly changing tuning values. The histogram is initialised with an appropriate mass γ to prevent rapid shifting of the mass centre in the beginning of tuning. An initial mass $\gamma = 10$ was used in this work.

The leak factor α controls the influence of previous pitch estimates to the tuning when accumulating the histogram values in Equation 3.8 and updating the histogram mass centre in Equation 3.9. If $\alpha \approx 1$, the histogram and the mass centre change slowly, and a longer history of pitch estimates affects the tuning. On the other hand, $\alpha \approx 0$ causes the algorithm to ignore previous pitch estimates, and the algorithm practically just rounds the pitch estimates to the nearest MIDI note numbers. In this work, a leak factor $\alpha = 0.99$ is used. Moreover, if the needed tuning is greater than the tuning grid values permit, the tuning grid is adjusted so that the histogram mass centre is near the centre of the tuning grid. This allows the tuning grid to be shifted as the tuning of a melody drifts, even more than a semitone.

The tuning algorithm is applied in the transcription system to tune pitch estimates given by the YIN algorithm. Throughout the thesis, pitch estimates are always tuned with the algorithm, and they are denoted with x. The algorithm is demonstrated in Figure 3.4, where panel (a) shows an estimated pitch curve and the reference melody. During the melody, the tuning drifts one semitone downwards. Subsequently, the pitch estimates are processed by the tuning algorithm, and the tuned pitches are shown in panel (b). Panel (c) shows the mass centre of the histogram and the tuning grid values as a shaded box, where the gradual shift of the tuning grid can be clearly seen. The drifting of the tuning has been compensated for by the algorithm and, during the process, the tuning grid shifts from values [-0.4, 0.5] to [0.7 1.6].

3.2 Accent and meter estimation

In addition to the tonal component, the rhythmic component of music provides valuable information concerning melodies. The rhythmic component analysis is based on *phenomenal accents* in music signals which indicate the moments in music containing perceptual emphasis [Lerdahl83]. In our transcription system, the degree of phenomenal accents is measured as a function of time, producing what we call an *accent* signal. In addition, a meter estimation system is applied and the results are used to estimate the degree of metrical accent as a function of time, resulting in what we call a *meter* signal. These are both used as musical features in the transcription system. The accent signal correlates with the time instants of performed note beginnings whereas the meter signal expresses the underlying musical meter of the performance. The accent and meter signals are produced by the methods proposed by Klapuri in [Klapuri03b] and discussed in the following sections.

3.2.1 Accent estimation

Phenomenal accents are here used to indicate the potential time instants of note beginnings, thus improving the melody-analysis accuracy. The transcription system utilises a method for estimating phenomenal accents from music signals as a front-end processor. The method was proposed by Klapuri in [Klapuri03b], and it is described and demonstrated in the following.

The method detects the signal intensity changes at the critical-bands of hearing, resulting in *registral accent signals.* These registral accent signals can be interpreted as measuring the degree of phenomenal accentuation in music signals. Since the method and its parameters are justified by the original author, the method is only briefly described in this thesis.

Given a discrete time-domain signal, the signal is first Hanning-windowed in successive 23 ms time frames which overlap by 50%. For each signal frame, the discrete Fourier transform is calculated and further divided into 36 critical bands by simulating triangular-response bandpass filters. The signal power at each frequency band

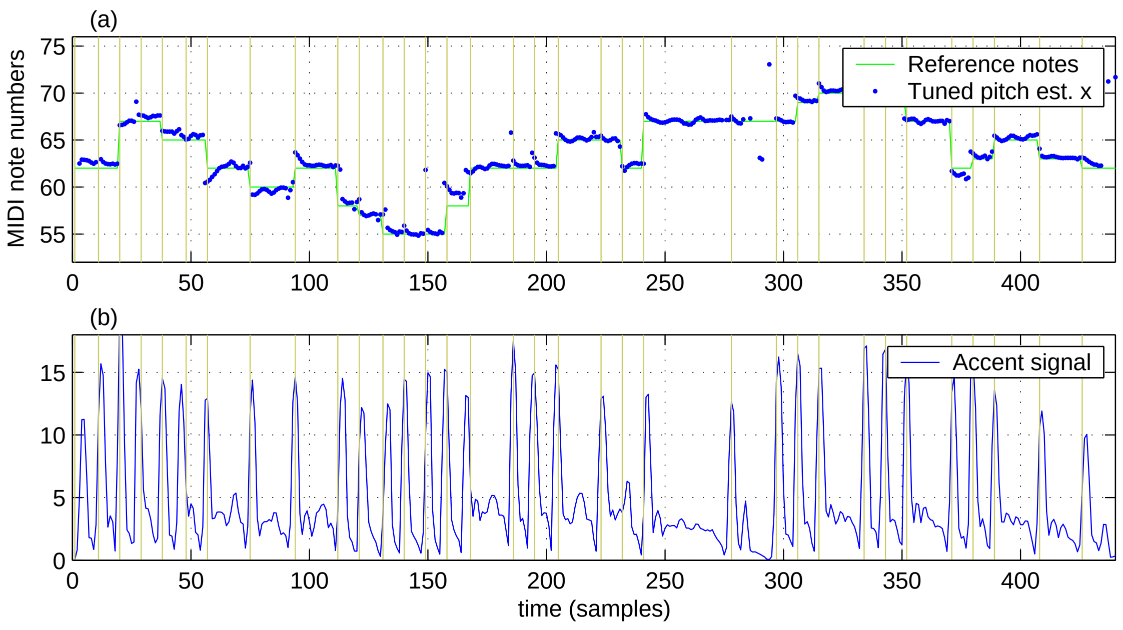

Figure 3.5: (a) A time-domain signal containing a melody, and (b) the corresponding accent signal $a(n)$. In this case, the accent peaks above the value 10 clearly indicate note boundaries.

$b \in \{1, 2, \ldots, 36\}$ is evaluated and denoted with $x_b(k)$ where k is the frame index. Subsequently, $x_b(k)$ is μ-law compressed,

$$y_b(k) = \ln[1 + \mu x_b(k)]/\ln(1 + \mu), \tag{3.11}$$

where μ affects the linearity of the mapping; in this work, the value $\mu = 100$ is used. Next, the compressed power envelopes $y_b(k)$ are interpolated by a factor of two to achieve a better time resolution, resulting in $y'_b(n)$ with sampling rate $f_r = 172$ Hz. During interpolation, the power envelopes are smoothed by a sixth-order Butterworth lowpass filter with $f_{LP} = 10$ Hz. The resulting signal is denoted by $z_b(n)$.

Since the method attempts to detect the *changes* in signal intensity, $z_b(n)$ is differentiated and then half-wave rectified

$$z'_b(n) = \mathrm{HWR}\{z_b(n) - z_b(n-1)\}, \tag{3.12}$$

where HWR$\{\cdot\}$ stands for half-wave rectification (i.e., negative values are mapped to zero). Further, $z_b(n)$ and its differential $z'_b(n)$ are averaged by

$$u_b(n) = (1 - w)z_b(n) + w\beta z'_b(n), \tag{3.13}$$

where $w = 0.9$, and the factor $\beta = f_s/(2f_{LP})$ compensates the small amplitude of the lowpassed-signal differential. The accent signal is here obtained by summing the frequency bands, and it is defined as

$$a(n) = \sum_{b=1}^{36} u_b(n). \tag{3.14}$$

Since the sampling rate of the accent signal (172 Hz) is greater than the musical feature sampling rate (1/0.025 s = 40 Hz, where 0.025 seconds is the musical-feature frame length), the maximum of the accent signal is used during each musical-feature period. In Klapuri's meter estimation system, the degree of accent is measured separately of four different frequency ranges, resulting in the registral accent signals. Here this is unnecessary and thus the summing in Equation 3.14 goes over all bands.

Figure 3.5 shows an example of an accent signal which has been calculated from a performed melody. Panel (a) shows a reference melody and the pitch estimate curve, and panel (b) shows the accent signal $a(n)$. Reference note beginnings are denoted by vertical lines. As shown in the figure, the clear accent signal peaks match with the note beginnings. In this case, peaks with values greater than 10 indicate note beginnings. However in this work, we do not apply any threshold value to indicate note beginnings but, instead, use the statistical distribution of the accent value in a probabilistic manner within the system. The absolute values of accent signals are affected to some extent, for example, by singing techniques.

3.2.2 Meter estimation

The above-discussed registral accents can be further analysed to produce the *musical meter*. The meter consists of pulses at three different time scales: measure, tactus, and tatum. Measures divide a music performance into musical segments, tactus defines the tempo of the performance, and tatums express the pulse and the smallest temporal unit of the piece.

Meter estimation is performed by a method proposed by Klapuri in [Klapuri03b]. The method uses the registral accents as an input to produce the time instants of measures, tactus, and tatums, which together constitute the meter. The method performs periodicity analysis of the registral accents by using a bank of comb-filter resonators. The filterbank is used to estimate the periods and the phases at the mentioned three time scales. The periods mean the time durations between successive beats, and the phases mean the time distances of beats from the beginning of the performance. This periodicity analysis is then followed by two probabilistic models. The models interpret the periods and the phases given by the filterbank to produce musically meaningful meter information. Particularly, the models apply musicological knowledge by considering the dependencies between the time scales and the temporal relationship between successive beats. Finally, the model outputs are combined to produce the meter, consisting of the time instants for each time scale. The meter-estimation method can be implemented in both causal or non-causal ways of which the former requires more time to become steady. Since the melodies to be transcribed are usually short, non-causal implementation was employed in our transcription system.

Coding of metrical information

The time instants of the meter information need to be transformed into a useful format for the transcription system. In this work, the time instants are used to gen-

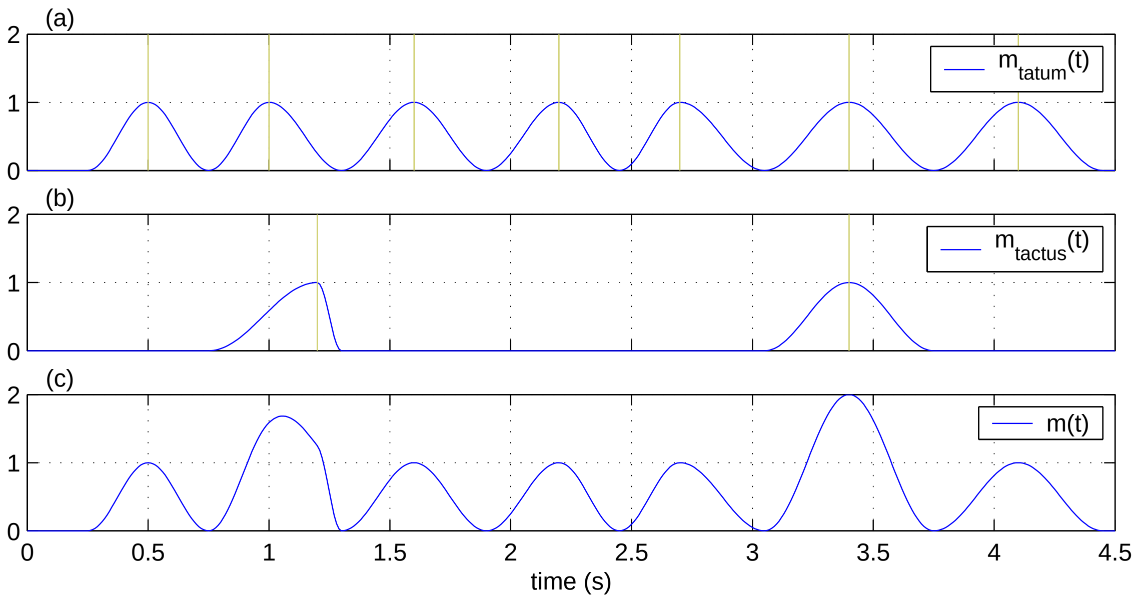

Figure 3.6: Meter signal $m(t)$ consisting of $m_{tatum}(t)$ and $m_{tactus}(t)$.

erate a meter signal $m(t)$ which can be considered as a musical feature representing metrical accentuation as a function of time t. Since the proposed transcription system is applied for short melodies, the measure estimation can not be performed very reliably, and therefore, only tactus and tatum are used to generate the meter signal.

The meter signal $m(t)$ is produced by generating sinusoidal pulses at those time instants where a tatum beat or a tactus beat has occurred. First, tatum beats are used to generate a signal $m_{tatum}(t)$ by

$$m_{tatum}(t) = \frac{1}{2}\left[\cos\left(2\pi\frac{t-\theta_i}{\theta_{i+1}-\theta_i}\right)+1\right], \qquad \theta_i \le t < \theta_{i+1}, \tag{3.15}$$

where θ_i is the occurrence time of the i:th tatum beat for $i = 2, 3, \ldots, T-2$, and θ_T is the last tatum beat time. If $i = 1$, the Equation 3.15 applies for $\theta_1 - (\theta_2+\theta_1)/2 \le t < \theta_2$ to prevent $m_{tatum}(t)$ from abruptly jumping to one at $t = \theta_1$. Correspondingly for $i = T-1$, the equation holds at $\theta_{T-1} \le t < \theta_T + (\theta_T + \theta_{T-1})/2$. Otherwise, $m_{tatum}(t)$ is zero.

Second, beats are used to generate a signal $m_{tactus}(t)$ in a similar fashion. However, tactus beats occur less frequently than tatum beats, and generating $m_{tactus}(t)$ like in Equation 3.15 would produce too wide sinusoidal pulses. Therefore, the pulse widths are determined by the tatum beats. Accordingly, given the k:th tactus beat time β_k, we search the tatum beat nearest to β_k and denote it with θ_i^k. The $m_{tactus}(t)$ is then defined as

$$m_{tactus}(t) = \begin{cases} \frac{1}{2}\left[\cos\left(2\pi\frac{t-\beta_k}{2\beta_k-\theta_i^k+\theta_{i-1}^k}\right)+1\right], & \frac{\theta_i^k+\theta_{i-1}^k}{2} \le t < \beta_k \\ \frac{1}{2}\left[\cos\left(2\pi\frac{t-\beta_k}{\theta_{i+1}^k+\theta_i^k-2\beta_k}\right)+1\right], & \beta_k \le t < \frac{\theta_i^k+\theta_{i+1}^k}{2} \\ 0, & \text{otherwise.} \end{cases} \tag{3.16}$$

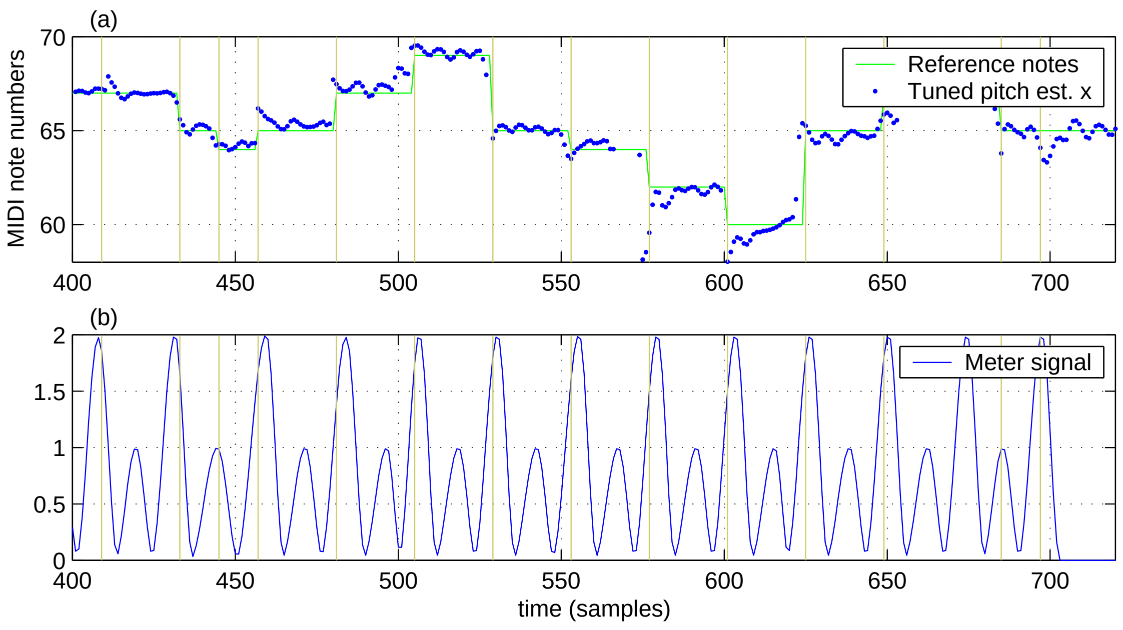

Figure 3.7: A meter signal extracted from a performed melody.

Finally, the two generated signals are combined to produce the meter signal

$$m(t) = m_{tatum}(t) + m_{tactus}(t). \tag{3.17}$$

The meter signal and its partials are illustrated in Figure 3.6, in which the tatum and tactus beat times are indicated by vertical lines. In particular, equations 3.15 and 3.16 produce identical pulses, if tactus and tatum beats are coincident. In the figure, this is clearly seen at time $t = 3.4$ s.

In the transcription system, the meter signal is used as a musical feature that predicts note onset times according to the underlying meter of the performance. As an example, Figure 3.7 shows an excerpt of a melody that has been performed while listening to the reference melody through headphones at the same time. The vertical lines denote the beginnings of the reference notes, and the reference notes and the tuned pitch estimates are illustrated in panel (a). According to the described method, meter information is extracted from the performance (without the accompaniment) and coded into the meter signal shown in panel (b). The meter signal contains high peaks whenever tactus and tatum beats have occurred simultaneously, and most of these peaks appear at the note beginnings. However, if melodies include longer notes (e.g., at the end of the example melody), the meter signal has peaks also during those notes. Nevertheless, the meter signal should not react to this type of sudden changes, since the meter represents the meter of the overall piece, not individual note beginnings.

4 Probabilistic Models for Melody Transcription

The stochastic nature of real-world signals motivates the use of *probabilistic methods* in signal-analysis problems. Despite the formality of musical representations, every musical performance involves errors, such as incorrectly performed notes in melodies, uncontrolled vibration in voice, and inaccuracies in timing. These errors must be handled to produce correctly transcribed melodies.

The proposed transcription system employs two probabilistic models: *a note model*, and *a musicological model*, both of which are based on the theory of *hidden Markov models* (HMM). The note model simulates musical features and their temporal behaviour within a note, thus modelling a meaningful musical unit. The musicological model, on the other hand, utilises the musical context and musicological knowledge to favour musically meaningful note sequences.

This chapter is organised as follows. Section 4.1 briefly reviews the theory of hidden Markov models, sections 4.2 and 4.3 introduce the note model and the musicological model, respectively. In Section 4.4, the proposed models are combined to constitute a probabilistic melody transcriber.

4.1 Hidden Markov models

In the beginning of the 20th century, Andrew A. Markov studied statistical properties of random processes and proposed a stochastic signal-model theory called the *hidden Markov model* (HMM). Hidden Markov models were further developed and published in a series of papers by L. E. Baum and colleagues in the 1960s and the 1970s. A few decades later, HMMs became a popular mathematical tool used in various applications. Particularly, pattern-recognition based applications, such as speech recognition systems, have widely exploited the theory of HMMs. The following review of HMMs is based on [Rabiner89], and a more practical use of HMMs is demonstrated in [Jelinek97, Duda01], for example.

Markov chains are random-process state machines that consist of three elements: a finite set of states $S = \{s_1, s_2, \ldots, s_K\}$, an initial state probability distribution, and a state-transition probability distribution. The set of states S defines the acceptable states of a random state sequence $q_1, q_2, \ldots, q_t$ in which $q_t \in S$ denotes a random state at time t. The initial state probability distribution assigns probabilities of the

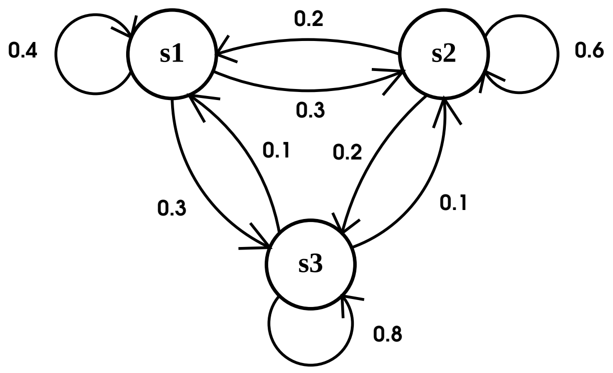

Figure 4.1: A Markov chain with a set of states $S = \{s_1, s_2, s_3\}$. The transition probabilities are indicated with directed arches. For example, $P(q_t = s_3|q_{t-1} = s_2) = 0.1$. The initial state likelihood distribution could be $P(q_1 = s_1) = 0.5$, $P(q_1 = s_2) = 0.2$, and $P(q_1 = s_3) = 0.3$, for example.

state machine to be in a certain state at time $t = 1$, i.e., $P(q_1 = s_i)$ where $s_i \in S$. The state-transition probability distribution defines the probabilities for transitions from state s_i to state s_j during the random process, $P(q_t = s_j|q_{t-1} = s_i)$. Figure 4.1 shows an example of a three-state Markov chain.

The mathematical power of Markov chains lies in the *first-order Markov assumption* stating that the random state q_t conditionally depends only of the preceding state q_{t-1} in a random-state sequence. This assumption may be formulated as follows [Papoulis84]:

$$P(q_t|q_1, q_2, \ldots, q_{t-1}) = P(q_t|q_{t-1}). \tag{4.1}$$

Furthermore, the conditional state dependency on *two* preceding states leads to the second-order Markov assumption, *three* to third-order Markov assumption, and so forth. In an alternative terminology, the state-probability dependency of $N - 1$ preceding states is also known as *N-gram approximation.*

Hidden Markov models are an extension of Markov chains. At time t, the state machine is observed through an observation vector $\mathbf{o}_t$ of the machine. However, it is not possible to directly observe the state that emitted the observation. Therefore, the relationship between the random state q_t and the corresponding observation needs to be modelled with an *observation likelihood* distribution $P(\mathbf{o}_t|q_t = s_i)$ where $s_i \in S$. This type of Markov chains are therefore called *hidden*, constituting a hidden Markov model.

As a result, the hidden Markov model can now be completely defined by the following parameters:

1. The set of states $S = \{s_1, s_2, \ldots, s_K\}$. The number of states is K.

2. The state-transition probabilities (for bigrams) $A = \{a_{ij}\}$ for which

$$a_{ij} = P(q_t = s_j|q_{t-1} = s_i), \qquad i, j \in \{1, 2, \ldots K\}. \tag{4.2}$$

The matrix A satisfies

$$\sum_{j=1}^{K} a_{ij} = 1, \qquad \forall\, i \in \{1, 2, \dots K\}, \tag{4.3}$$

because the probabilities to move from a state to any state must sum to unity. Analogous formulas can be written for trigrams as

$$a_{hij} = P(q_t = s_j | q_{t-2} = s_h, q_{t-1} = s_i), \qquad h, i, j \in \{1, 2, \dots K\} \tag{4.4}$$

and

$$\sum_{j=1}^{K} a_{hij} = 1, \qquad \forall\, h, i \in \{1, 2, \dots K\}. \tag{4.5}$$

3. The observation likelihood distribution $B = \{b_j(\mathbf{o}_t)\}$ for each state, where

$$b_j(\mathbf{o}_t) = P(\mathbf{o}_t | q_t = s_j), \qquad j \in \{1, 2, \dots K\}, \tag{4.6}$$

denotes the likelihood of observing $\mathbf{o}_t$ given the state s_j.

4. The initial state probability distribution $\boldsymbol{\pi} = \{\pi_i\}$, where

$$\pi_i = P(q_1 = s_i), \qquad j \in \{1, 2, \dots K\} \tag{4.7}$$

denotes the *a priori* likelihood of the state s_i being the initial state, i.e., the actual state at time $t = 1$. Similarly, this satisfies

$$\sum_{i=1}^{K} \pi_i = 1. \tag{4.8}$$

The complete parameter set of a hidden Markov model is conventionally denoted by $\lambda = (A, B, \boldsymbol{\pi})$. As an important remark, the observation likelihood distribution is occasionally defined as an observation-symbol probability distribution in which continuous observation vectors $\mathbf{o}_t$ are replaced with discrete observation symbols. In that case, the observation symbols are typically included in the set of HMM parameters. In the following presentation of HMMs, however, the observation vectors are continuous-valued and likelihood distributions are thoroughly used.

4.2 Note event model

In general, the most fundamental units in music are *notes* which constitute melodies, chords, and harmonies and thus the content of a music performance. Consequently, the importance of notes as musical units should be taken into account in melody transcription systems. However, the current state-of-the-art systems, including [Clarisse02, Viitaniemi03, Pollastri02], usually treat notes only at the level of individual signal frames, or, segment note events during a pre-processing stage, thus

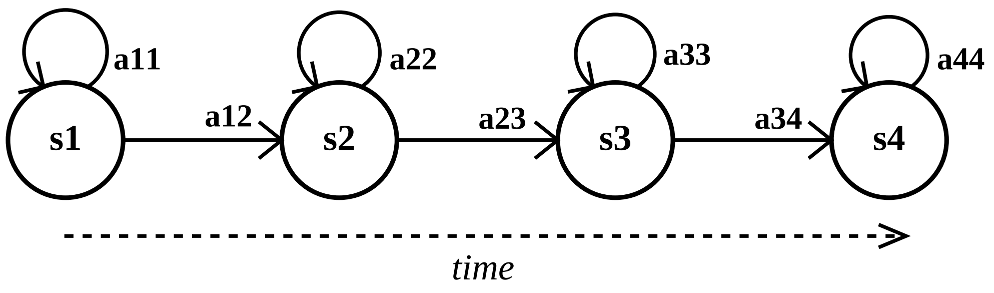

Figure 4.2: A four-state, left-to-right note event model (without skips).

ignoring the importance of notes as musical units or leading to a rigid segmentation of notes in time.

This thesis proposes a dynamic note event model based on HMMs. The proposed note model has two main advantages over the conventional approaches. First, musical features and their temporal behaviour are modelled with a probabilistic note event model, providing a dynamic and flexible interpretation of notes as musical units. Second, using notes instead of pitched frames as the basic analysis units enables a more profound melody analysis, since the musicological relationships between these units are well-defined when examining them as constituents of melodies and musical context.

The structure of the note event model is illustrated in Figure 4.2 where a four-state HMM is used to model a note event. The states of the model represent the temporal segments of note events in time. In other words, each state of the model imitates the typical behaviour of musical features at different time instants in the note event. For example, there typically exists some fluctuation in pitch at the beginning of notes, silence or noise at the end of notes, and so forth. As a result, the note model considers these peculiarities of the musical features as an integral part of the musical performance. In addition, the note-model states are not bound in time because it is possible to move from a state to itself. In Figure 4.2, these transitions are indicated with arches $a_{ii}, i \in \{1, 2, 3, 4\}$. Moreover, the note model is a *left-to-right* HMM without skips, i.e., it is possible to move from a state only to itself or to the following state. For this reason, there are only arches a_{ij} for which $j = i$ or $j = i + 1$. This is an important restriction, because the note model must undergo through all its states to completely model the typical behaviour of note events. Otherwise, the model could be executed by visiting only the first and the last state, possibly treating musically meaningless (e.g., noisy) segments as note events.

This section is organised as follows. Section 4.2.1 defines the HMM which is used to model note events, and Section 4.2.2 explains the training of the note model and introduces feasible parameter setups for the proposed note model. Finally, Section 4.2.3 explains the usage of the note model.

4.2.1 Note event HMM

The fundamental idea of the note HMM is to consider a note event as a state machine that emits musical features. While observing these musical features, the note state machine can be assumed to be in a certain state with a probability defined by the likelihood functions of note model. Hence, the extracted musical features represent an observation vector $\mathbf{o}_t$ that contains the features emitted by the state machine at time t. Since each musical feature is described by a single value, the length of the observation vector is equal to the number of features used. Furthermore, the observation likelihood distribution $b_j(\mathbf{o}_t)$ defines the likelihood that state s_j has emitted the observation vector $\mathbf{o}_t$.

The parameters of the note HMM are the following:

1. The set of states $S = \{s_1, s_2, \dots, s_K\}$ within the note model.
2. The state-transition probability distribution matrix $A = \{a_{ij}\}$, $A \in \mathbb{R}^{\mathrm{KxK}}$, where $a_{ij} = P(q_t = s_j | q_{t-1} = s_i)$ with $i, j \in \{1, 2, \dots, K\}$.
3. The observation likelihood distribution $B = \{b_j(\mathbf{o}_t)\}$ in which $s_j \in S$ and $b_j(\mathbf{o}_t) = P(\mathbf{o}_t | q_t = s_j)$.
4. The initial state probability distribution $\boldsymbol{\pi} = \{\pi_i\}$ for which $\pi_1 = 1$, and $\pi_i = 0$ for $i \neq 1$.

The note-model states define the temporal accuracy of the note model; the more there are states in the note model, the more accurate is the temporal separation of musical features during note events. However, the number of states should be adjusted to preserve both the accuracy and the generality of the note model. In this work, note models with one to five states are simulated.

The state-transition probability distributions define the probabilities for moving from a state to another. Conventionally, these probabilities are represented with a square matrix called a *state-transition matrix* $A \in \mathbb{R}^{\mathrm{KxK}}$. Since the note model is left-to-right HMM without skips, $a_{ij} \neq 0$ only for $j = i$ and $j = i + 1$. Particularly, if $a_{ij} = 0$, a transition from state s_i to state s_j is not possible.

The observation likelihoods B define the likelihoods that a certain state of the model has emitted the observation vector. Each state $s_j \in S$ has a corresponding C-variate probability distribution modelled with a Gaussian mixture model (GMM) (i.e., a weighted sum of η Gaussian densities). Here the number of used musical features is C and corresponds to the length of observation vector $\mathbf{o}$. Now $P(\mathbf{o}|s_j)$ gives the likelihood that an observation vector $\mathbf{o}$ is emitted by the state s_j:

$$b_j(\mathbf{o}) = \sum_{i=1}^{\eta} \frac{w_{i,j}}{(2\pi)^{C/2} |\boldsymbol{\Sigma}_{i,j}|^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{o} - \boldsymbol{\mu}_{i,j})^T \boldsymbol{\Sigma}_{i,j}^{-1} (\mathbf{o} - \boldsymbol{\mu}_{i,j}) \right], \tag{4.9}$$

where $w_{i,j}$ is the component weight, $\boldsymbol{\mu}_{i,j}$ is the C-dimensional mean vector, and $\boldsymbol{\Sigma}_{i,j}$ is the covariance matrix of the i:th GMM component of the observation likelihood distribution of the state s_j. Moreover, the covariance matrices are assumed to be diagonal matrices in this work, meaning that the observation-vector elements are assumed to be statistically independent of each other [Duda01, p. 34]. Therefore, the observation likelihoods for a state can be calculated by simply multiplying the distributions of each individual feature. Given that o_i is the i:th element of the observation vector $\mathbf{o}$, and $b_{j,i}(o_i)$ is the observation distribution of the i:th feature for state s_j, the overall likelihood can be defined as

$$b_j(\mathbf{o}) = \prod_{i=1}^{C} b_{j,i}(o_i). \tag{4.10}$$

Finally, the initial-state distribution $\boldsymbol{\pi}$ simply states that the note-model Markov sequence always starts from the first note-model state s_1.

4.2.2 Note model training and parameters

In order to model real-world note events, the note model should be trained with an acoustic database. The acoustic database used in this work was created in [Viitaniemi03] containing approximately 120 minutes of audio material performed by five male and six female non-professional singers. The singers performed four folk songs and two scales with different techniques, including singing, humming, and whistling. The singers were accompanied by MIDI representations of the melodies which they heard through headphones while performing. Only the performed melody was recorded and, later, the reference accompaniments were synchronised with the performances. On the whole, the database contains over 9000 performed notes. Eventually, the performances of three male and four female singers (excluding the whistling performances and the scales) were used to train the note model, constituting the training set of the model. This training set includes approximately 3100 note events in total.

The training set contains acoustic performances that are processed in 25 ms frames to extract the musical features discussed in Chapter 3. In addition to the musical features, the synchronised MIDI accompaniments include *reference notes* associated with the time frames in which the features are extracted. Particularly, the reference notes are essential in determining the *difference* between the reference notes and the performed pitch contour during note events. This difference x_Δ is defined as

$$x_\Delta = x - x_{\text{ref}} \tag{4.11}$$

where x is the tuned pitch estimate, and x_{ref} is the reference note in the MIDI accompaniment.

The reference notes are also used for segmenting the training data into distinct note events. A single note event is considered to begin from a reference-note onset and to end at the onset of the subsequent note in the reference melody. A segmented note event determines the time frames where the extracted features correspond to that note event. Figure 4.3 illustrates the note-event segmentation for an excerpt

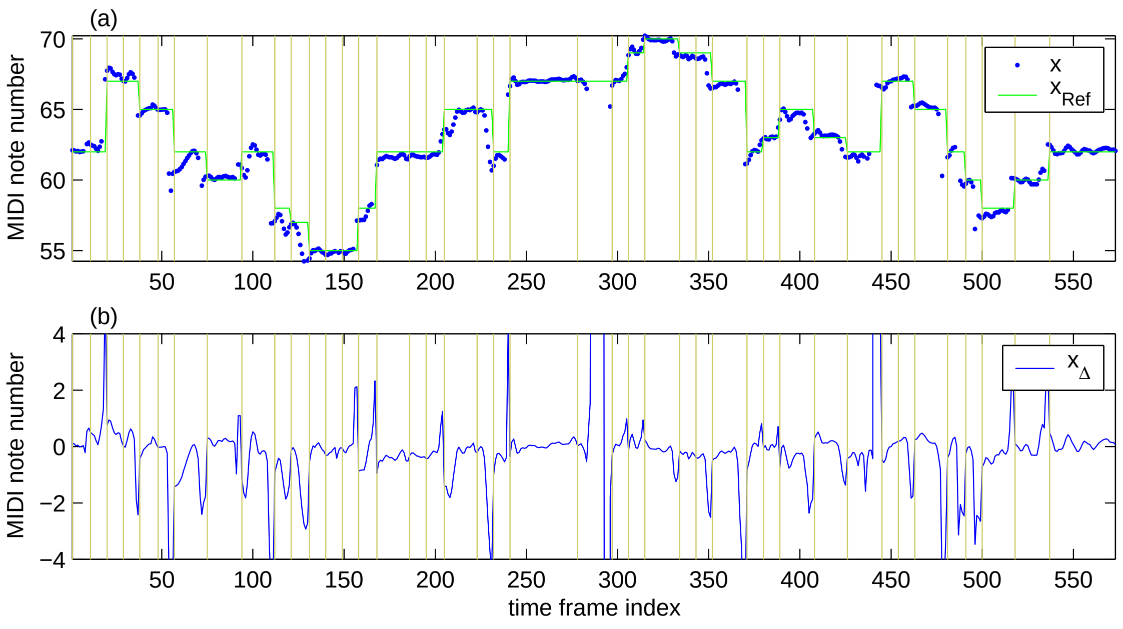

Figure 4.3: Segmentation of the training data according to the reference notes: (a) an estimated pitch curve and the reference notes and (b) the values of $x_\Delta = x - x_{\text{ref}}$.

from the acoustic database. In addition, panel (b) shows the difference between the reference notes and the performed pitch contour, x_Δ.

After the segmentation, the note-HMM parameter set λ must be estimated so that the note HMM models note events as accurately as possible. The parameter estimation is performed using the *Baum-Welch algorithm* (or the *expectation-maximisation* (EM) algorithm) which provides a maximum-likelihood estimate for the HMM parameters. The algorithm iteratively searches the parameter set λ that locally maximises the likelihood of observation sequences (consisting of the extracted musical features) within note events given the note-model parameter set. In other words, the algorithm maximises the likelihood $P(O|\lambda)$ where O denotes all the observation sequences during note events. The Baum-Welch algorithm is a well-known method, and it has been described by several authors, for example in [Rabiner89, pp. 264-266]. Therefore, the details of the algorithm are not represented here.

The training of the note model results in a complete parameter set λ for the model. The parameter set includes the state-transition probabilities A and the observation likelihoods B. Particularly, B distribution is specified by Gaussian mixture models as given by Equation 4.9.

Estimated model parameters

The note model includes several adjustable parameters which affect its performance. Among these parameters are the number of states K, the number of Gaussian components η for each observation likelihood distribution, and the set of musical features

Parameter	Values
number of note-model states, K	1, 2, 3, 4, 5
number of GMM components, η	1, 2, 3, 4, 5, 6
feature set	$\{x_\Delta, \nu\}, \{x_\Delta, \nu, a\}, \{x_\Delta, \nu, m\}, \{x_\Delta, \nu, a, m\}$

Table 4.1: The note HMM training parameters.

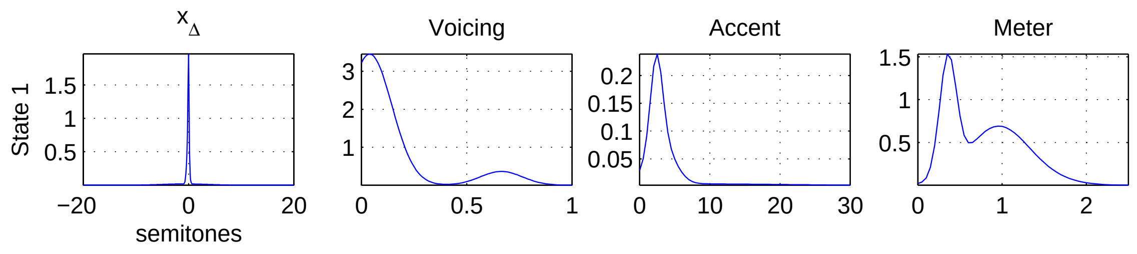

Figure 4.4: Note-HMM observation likelihoods B with $K = 1$, $\eta = 4$.

used. In addition, each musical feature can be *weighted* according to its significance. In practise, this weighting determines the influence of each musical feature to the observation-likelihood distribution. However, the weighting can be performed after the note-model parameters have been estimated, and therefore, feature weighting is discussed later in Section 4.2.3.

The note model was trained with the parameters represented in Table 4.1. The number of model states and Gaussian components in the observation likelihood distribution varied from one to five and from one to six, respectively. In addition, four different sets of musical features were used: $\{x_\Delta, \nu\}$, $\{x_\Delta, \nu, a\}$, $\{x_\Delta, \nu, m\}$, $\{x_\Delta, \nu, a, m\}$, in which x_Δ, ν, a, and m denote the difference between the tuned note estimates and the reference notes, voicing, accent, and the metrical accent, respectively.

The following examples illustrate the effect of the parameters K and η to the observation likelihoods B. The note models have been trained with the musical feature set $\{x_\Delta, \nu, a, m\}$, and figures 4.4-4.8 show the distributions B for K values 1, 2, 3, 4, and 5, respectively. First, Figure 4.4 shows the observation likelihood distributions for a single-state model for which each feature distribution is modelled with four GMM components. Since the model consists of only one state, the distributions show the typical values of each musical feature during note events, i.e., x_Δ is approximately zero, voicing and accent get small values, and meter distribution has two peaks at 0.5 and 1 that correspond to the mean values of the sinusoidal signals produced by tatum and tactus, respectively.

In a two-state model (see Figure 4.5), the second state clearly models the distribution of the musical features at the note-event endings: x_Δ and a values disperse and the distribution of voicing becomes bimodal in the second state of the note, thus modelling *silence* or *noise* at the note endings. The bimodal distribution is a consequence of having rests (i.e., silence) at the end of some notes. The meter values are almost equally distributed in both note-model states.

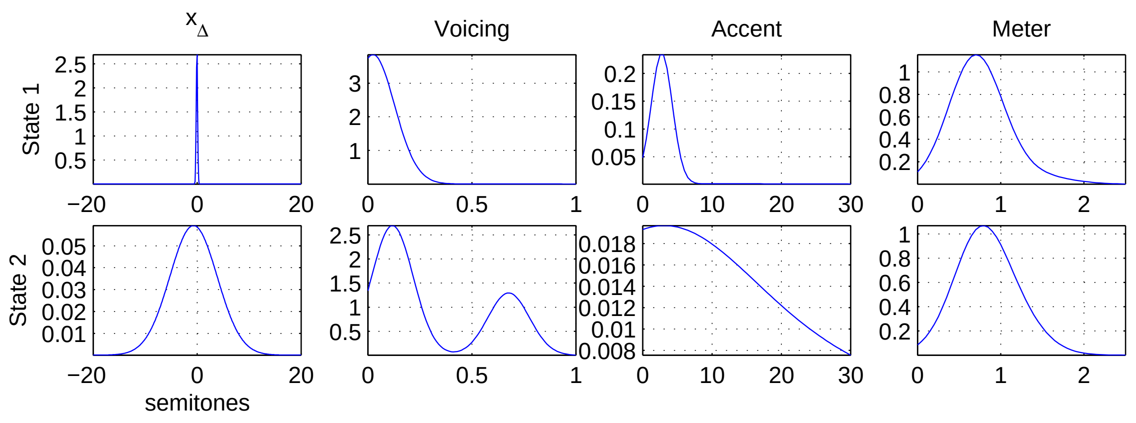

Figure 4.5: Note-HMM observation likelihoods B with $K = 2$, $\eta = 2$.

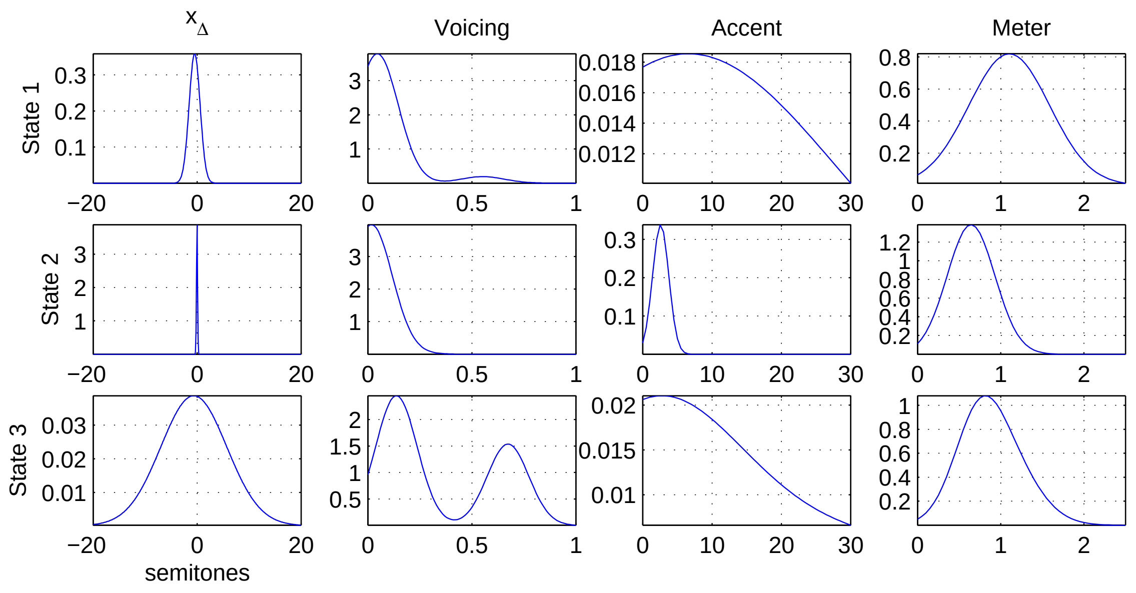

Figure 4.6: Note-HMM observation likelihoods B with $K = 3$, $\eta = 2$.

The most intuitive feature distributions are introduced by a three-state note model in Figure 4.6. The note-model states 1, 2, and 3 could be interpreted as *transient*, *sustain*, and *silence* segments of a note event, respectively. The transient segment of note events results in a wide-spread accent-value distribution, typical meter values near one expressing the occurrence of tatums, and some variance in the values of x_Δ. During the sustain stage, the variance of x_Δ becomes smaller, the frames are mostly voiced ($\nu < \kappa$, κ is the threshold value 0.15), and the accent and meter values are smaller. Eventually in the silence state, x_Δ values spread, and voicing becomes bimodal. This shows that some of note events include silence or noise at the end of the events, usually as a consequence of breathing or consonants between notes. Accordingly, these distributions express the predictable behaviour of the musical features during note events. In fact, the note model with three states and two GMM components performed best in the simulations.

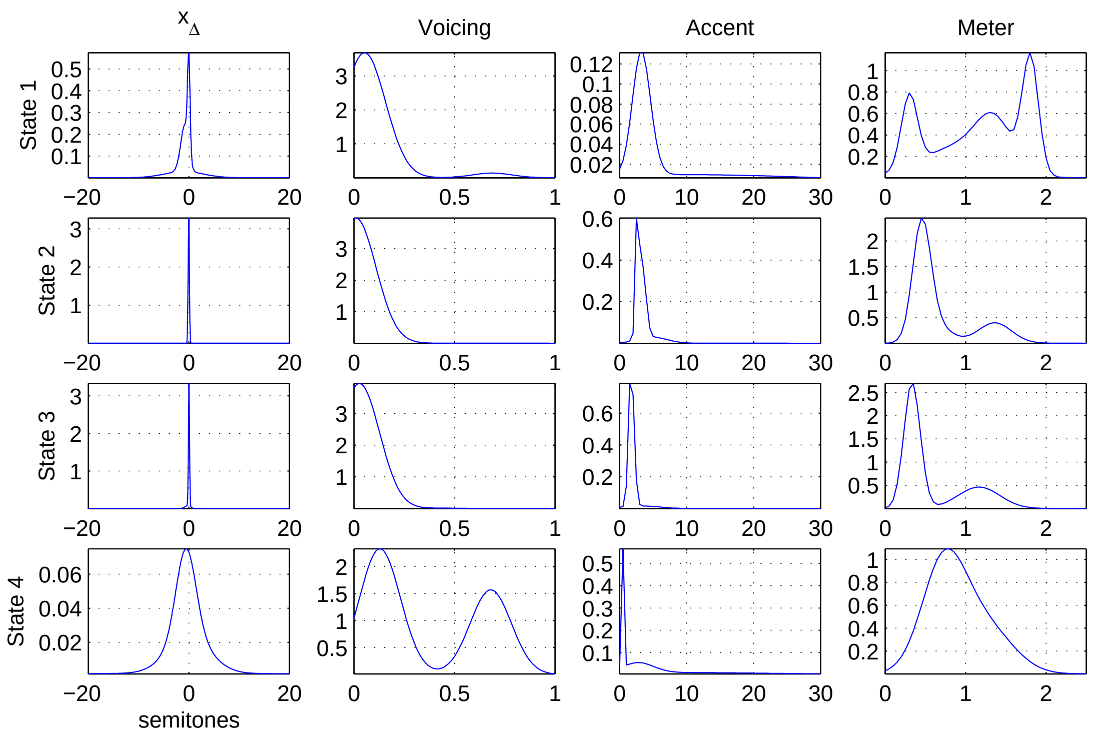

Figure 4.7: Note-HMM observation likelihoods B with $K = 4$, $\eta = 5$.

The four-state and five-state note models are presented in Figures 4.7 and 4.8, respectively. Although these models have a greater temporal accuracy, the states within the models resemble each other too much, resulting in redundant states. For example in Figure 4.8, the note-model states 2-4 significantly resemble each other in observation distributions. As an exception, meter has a high-valued peak in the third state in Figure 4.8, suggesting that tatum and tactus beats have simultaneously occurred also in the middle of note events as expected. Note HMMs with different feature sets could benefit from the temporal accuracy, but the proposed feature sets worked best with three or four states in the model.

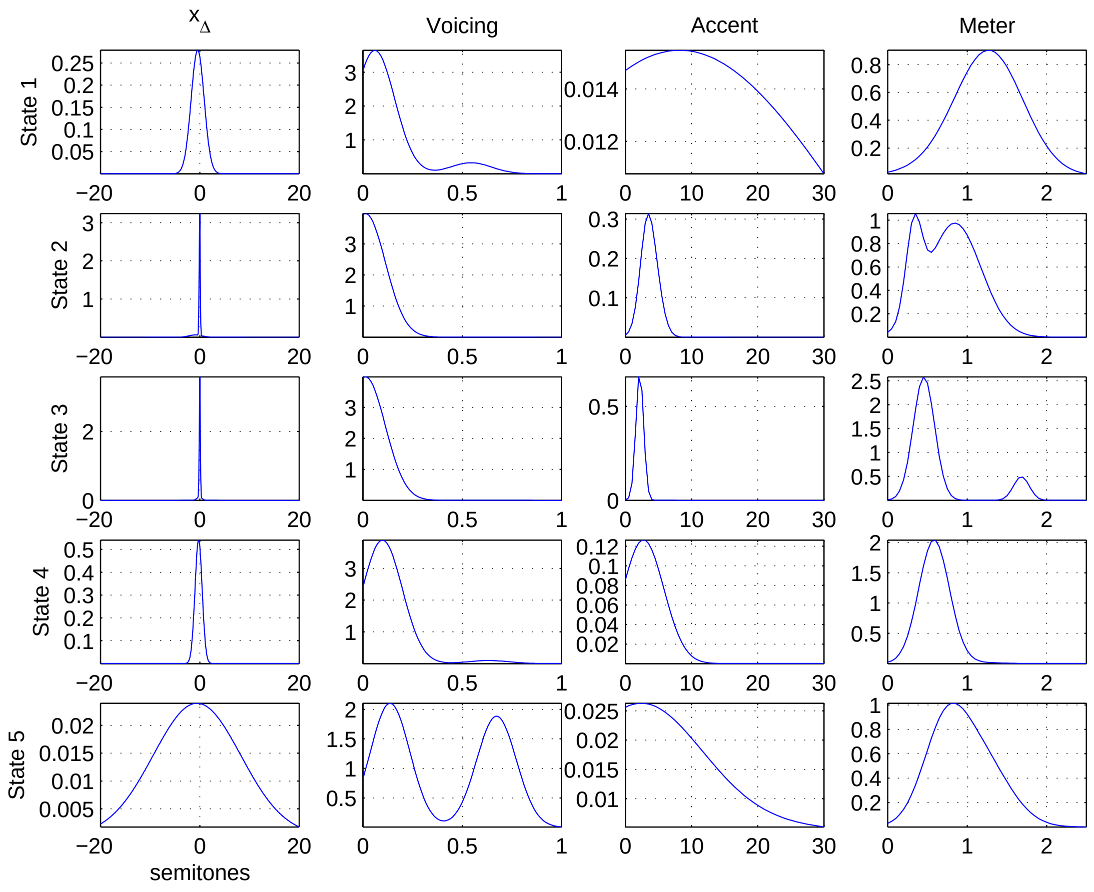

Figure 4.8: Note-HMM observation likelihoods B with $K = 5$, $\eta = 2$.

4.2.3 Using the note model

The note models are used to represent different notes in the transcription system. Specifically, there exists one note model per each MIDI note number n (e.g., with values 36 to 96). Since the note-event models were trained by using the difference x_Δ between the tuned pitch estimate x and the reference MIDI note x_{ref}, the note models must be used accordingly. In other words, the note models in the transcription system must apply the MIDI note numbers n instead of the reference MIDI note numbers x_{ref} to calculate the x_Δ. In particular, the other musical features (voicing, accent, and meter) are not affected by MIDI note values, which has the consequence that the contribution of those features is equal for every note model in the transcription system. In addition, also the other note-model parameters, such as the state-transition probabilities A and the initial state distribution $\boldsymbol{\pi}$, are independent of the MIDI-note value.

Given that $s_{j,n}$ is the j:th state of the note model representing a MIDI note n, the tuned pitch estimate x in the observation vector $\mathbf{o}$ must be replaced with the x_Δ defined by

$$x_\Delta = x - n. \tag{4.12}$$

Subsequently, the observation likelihood $P(\mathbf{o}|s_{j,n})$ defines the likelihood to observe the modified observation vector $\mathbf{o}$ given that it was emitted by the j:th state of the note model n. The observation likelihood can be calculated by the Equation 4.9 with the estimated note-model parameters.

The observation-likelihood calculations can be performed very efficiently, since voicing, accent, and meter have an equal contribution to each note model. In other words, the observation likelihoods need to be calculated only for a single note model (with these features) at each time, and then the values can be copied to the other note models. In addition, the distributions can be represented as cumulative distribution functions from which the likelihood values can be obtained very efficiently.

Musical feature weighting

The influence of different musical features can be adjusted by *weighting* their contribution in the observation-likelihood distribution. Equation 4.10 defined the likelihood by using the features of the observation-likelihood distribution. Now the equation is reformulated as

$$b_j(\mathbf{o}) = \prod_{i=1}^{C} b_{j,i}(o_i)^{w_f(i)}, \tag{4.13}$$

where $w_f(i)$ denotes the weight for the i:th feature. The used feature weights were empirically determined by examining the influence of each feature to the transcription performance. The pitch difference (i.e., x_Δ) was not weighted at all, and voicing ν, accent a, and meter m were weighted by a factor 10. As an exception, meter was not weighted when using all the four features, i.e., the feature set $\{x_\Delta, \nu, a, m\}$.

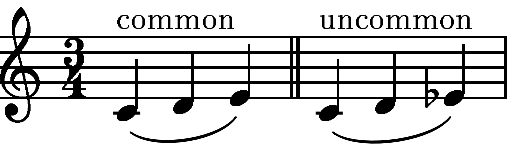

Figure 4.9: Examples of a common and an uncommon three-note sequences in the C-major-key context.

4.3 Musicological model

Despite the variety of musical genres, there are general musical constraints or rules that govern the Western music. These rules may vary according to music styles (or even composer) and they affect, e.g., the harmony, melodies, and rhythms of musical pieces to an important degree. Therefore, *musicological modelling* enables a more profound approach to music-content analysis and improves the robustness of music analysis systems. Musicological modelling refers to a set of musicological models exploited in the music-content analysis, and each musicological model is composed of musicological rules, constraints, and assumptions that constitute a well-defined musicological entity.

The proposed transcription system applies a combination of two probabilistic musicological models: a musical-key estimation model and a note-sequence likelihood model. The models are combined so that the note-sequence model assigns likelihoods for N-note sequences in a certain *tonal context* which is defined by the key-estimation model. In other words, the musicological model considers certain note sequences to be more common than others in a certain tonal (musical) context. Although the musical context of a song is affected by several elements, the musical key determines the main tonality of the entire piece. Consequently, ambiguous note sequences can be solved to satisfy common musical conventions in a probabilistic manner, thus leading to a transcription which both sounds musical and was presumably striven by the performer.

The influence of the tonal context on note sequences is illustrated in Figure 4.9 in which two almost identical three-note sequences are represented in the C-major-key context. Both sequences start with notes C and D followed by notes E and E♭. Despite only a semitone difference in the last notes, the first sequence is significantly more common than the second one. Given the key of C major and an ambiguous note estimate between the notes E and E♭, the note-sequence model would prefer the note sequence C, D, E, thus producing a conventionally transcribed note sequence. However, the note sequence C, D, E♭ would be more preferable if the musical key was C *minor* instead of C major.

In the following presentation, *pitch class* representation of notes is frequently used. Pitch classes express the octave equivalence of notes, i.e., tones separated by an octave (or several octaves) are considered to be musically equivalent in many ways [Burns99, p. 252]. Hence, given a MIDI note number n, the corresponding pitch class n^{pc} is defined by

$$n^{pc} = \mathrm{mod}(n, 12). \tag{4.14}$$

k_{maj}	Key	k_{min}	Key
0	C major	0	C minor
1	D♭ major	1	C♯ minor
2	D major	2	D minor
3	E♭ major	3	E♭ minor
4	E major	4	E minor
5	F major	5	F minor
6	F♯ major	6	F♯ minor
7	G major	7	G minor
8	A♭ major	8	G♯ minor
9	A major	9	C minor
10	B♭ major	10	B♭ minor
11	B major	11	B minor

Table 4.2: The relationship between tonic notes and musical keys.

For example, if the C major scale is defined by the MIDI notes $60, 62, 64, 65, 67, 69, 71$ or by the notes $48, 50, 52, 53, 55, 57, 59$. Both definitions are equivalent according to pitch class representation $0, 2, 4, 5, 7, 9, 11$. Specifically, pitch class values $0, 1, \ldots, 11$ correspond to notes C, C♯, D, …, B.

This section is organised as follows. Section 4.3.1 explains the key-estimation model, and Section 4.3.2 defines the note-sequence model based on the likelihoods of N-note sequences in the context of different musical keys. Finally, Section 4.3.3 combines the models to constitute the musicological model used by the transcription system.

4.3.1 Key estimation

Melodies are conventionally performed in a musical key which defines the basic seven-note major or minor scale used throughout a song. The key of the song affects significantly to the occurrence frequencies of different notes during the song. Particularly, most of the notes in melodies belong to the scale determined by the key. Conversely, the likelihoods of keys can be determined based on the notes which occur in melodies. Further, the estimated key likelihoods can be used to examine the likelihoods of notes in melodies meaning that exceptional note occurrences (i.e., notes not belonging to the key) can be detected. This relationship between notes and keys can be therefore applied in music-content analysis.

In this work, musical keys are defined by major and minor *tonic notes* $k_{maj}, k_{min} \in \{0, 1, \ldots, 11\}$ so that k_{maj}, k_{min} values $0, 1, \ldots, 11$ correspond to the major and minor keys with tonic notes C, C♯, D, …, and B. Table 4.2 shows the relationship between tonic notes and musical keys. In addition, each major key defines a scale that consists of the same seven notes as the *relative* minor key. For example, both the C major scale and the A minor scale consist of the same notes, but the tonic notes are different (note C for C major, and note A for A minor). The relationship between relative major and minor keys can be expressed by using tonic notes as follows:

$$k_{maj} = \mathrm{mod}(k_{min} + 3, 12) \quad \Leftrightarrow \quad k_{min} = \mathrm{mod}(k_{maj} + 9, 12). \tag{4.15}$$

As an example, the relative tonic note for D minor (i.e., $k_{min} = 2$) is $k_{maj} = 5$ according to Equation 4.15, thus addressing the *relative-key pair* of F major and D minor.

Key estimation is applied in this work to determine the tonic notes of the relative-key pairs of the melodies to be transcribed. Estimation is performed by a method proposed by Viitaniemi *et al.* in [Viitaniemi03] with few simplifications. The method estimates the likelihoods of keys for given pitch estimates $X = [x_1, x_2, \ldots, x_T]$ where x_t denotes a tuned pitch estimate at time t. The calculations are greatly simplified by rounding the pitch-estimate values to the nearest integers and expressing them as pitch classes n^{pc},

$$n_t^{pc} = \mathrm{mod}(\langle x_t \rangle, 12), \qquad t = 1, 2, \ldots, T, \tag{4.16}$$

thus producing a pitch-class sequence $X = [n_1^{pc}, n_2^{pc}, \ldots, n_T^{pc}]$. The simplified method is sufficiently robust for our purposes, and it is briefly explained in the following.

The key estimation is based on the occurrence likelihoods of different pitch classes n^{pc} in given keys k, i.e., $P(n^{pc} = i|k = j)$ where k is a major or minor tonic note. The occurrence likelihoods have been estimated by several authors, and the likelihoods estimated in [Krumhansl90, p. 67] are applied in this work. By using the Bayes formula, the likelihoods $P(n^{pc} = i|k = j)$ can be used to calculate the likelihoods of keys given notes. Thus, the probability of the j:th key given a pitch class $i \in \{0, 1, \ldots, 11\}$ is

$$P(k = j|n^{pc} = i) = \frac{P(n^{pc} = i|k = j)P(k = j)}{P(n^{pc} = i)}, \tag{4.17}$$

which may be reduced to

$$P(k = j|n^{pc} = i) \propto P(n^{pc} = i|k = j), \tag{4.18}$$

since the a priori probabilities of pitch classes $P(n^{pc} = i)$ and keys $P(k = j)$ are assumed to be uniform over i and j, respectively. Finally, the probability of the j:th key given the sequence $X = [n_1^{pc}, n_2^{pc}, \ldots, n_T^{pc}]$ is given by

$$P(k = j|X) = \prod_{t=1}^{T} P(k = j|n_t^{pc}). \tag{4.19}$$

The usage of the method is demonstrated by the following example. Figure 4.10 represents the note-occurrence likelihoods from [Krumhansl90, p. 67]. This likelihood distribution expresses the likelihood $P(n^{pc} = i|k = j)$ as follows; given a pitch class n^{pc} and a tonic note k, the distance of a pitch class n^{pc} from a tonic note k (i.e., $\mathrm{mod}(n^{pc} - k, 12)$) determines the likelihood for major and minor keys in the distribution. For example, given a note B ($n^{pc} = 11$) and E minor key $k_{min} = 4$, the distance

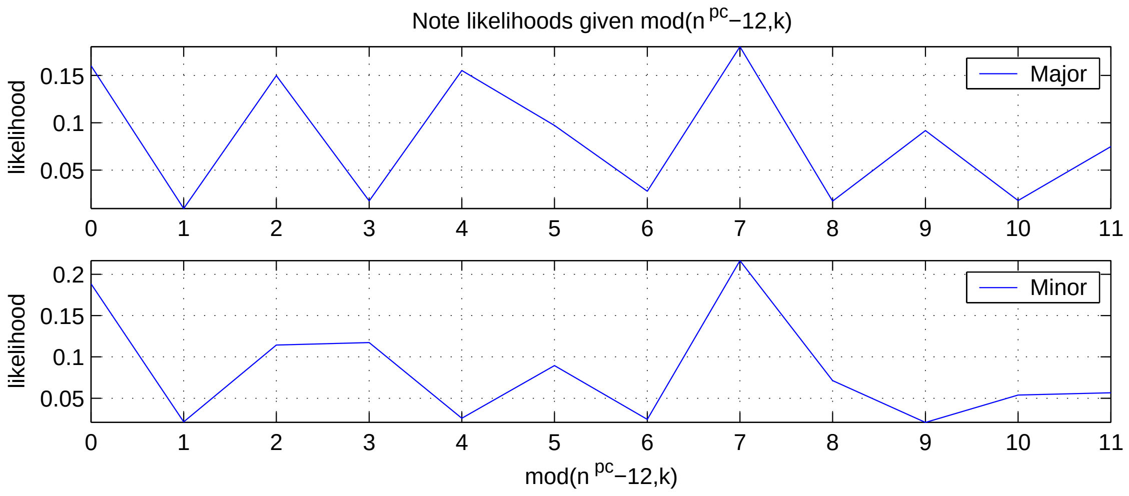

Figure 4.10: The likelihoods of pitch classes given key.

$\mathrm{mod}(11-4,12)=7$ results in a likelihood value $P(n^{pc}=11|k_{min}=4)\approx 0.22$ from the distribution.

Considering a pitch-class sequence $X=[n_1^{pc},n_2^{pc},\ldots,n_T^{pc}]$, the likelihoods of each key are determined by Equation 4.19, in which the keys are expressed as tonic notes $k_{maj},k_{min}\in\{0,1,\ldots,11\}$. Further, the likelihoods of relative major and minor keys are summed together so that the relationship in Equation 4.15 is obeyed. This produces a likelihood distribution of relative-key pairs for the given note sequence. Consequently, the maximum of the joint distribution determines the most probable relative-key pair. The key estimation is used non-causally in the transcription system, i.e., the most probable relative-key pair is first determined based on all the pitch estimates, and after that the results of key estimation are applied.

4.3.2 Note-sequence likelihoods

Melodies are basically comprised of note sequences that are more or less common in certain musical contexts. To be more precise, the occurrence likelihoods of note sequences are affected by the tonality of a song. Clearly, also the culture and musical genre orientations affect the listeners' opinion on what type of note sequences sound *musical*. However, each music style can be characterised by the note sequences that are most often used in melodies of that music style. The likelihoods of note sequences can be estimated from melody databases by counting the occurrence rates of different note sequences.

A sequence of N notes is represented in this work by using two elements: a tonic distance d, and a vector of directed intervals i_t. The tonic distance defines the distance between a note n and the tonic note k,

$$d=\mathrm{mod}(n-k,12), \tag{4.20}$$

where tonic note determines the key of a song. For an N-note sequence, $(n_{t-N+1}$, $n_{t-N+2},\ldots,n_t)$ the tonic distance is calculated only for the *first* note of the sequence,

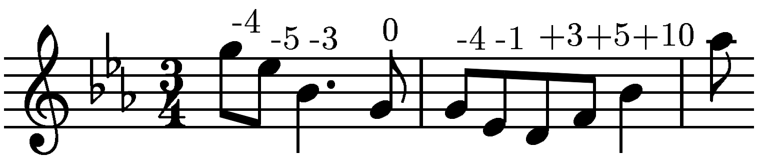

Figure 4.11: An example melody in E♭ major for the note-sequence likelihood estimation.

$$d_t = \mathrm{mod}(n_{t-N+1} - k, 12). \tag{4.21}$$

Second, the directed interval i_t at time t is defined as

$$i_t = n_t - n_{t-1}, \tag{4.22}$$

by which the note-sequence-interval vector may be defined as $[i_{t-N+2}, \ldots, i_t] \in \mathbb{Z}^{\mathrm{1x(N-1)}}$.

The proposed melody-transcription system defines a musicological model using likelihood estimates for N-note sequences. The note-sequence likelihoods for $N \in \{2, 3\}$ are estimated from EsAC (Essen Associative Code and Folksong Database) database [Dahlig94] currently containing over 20000 folksongs mainly from Europe and China. Approximately 6000 of these songs were used in the note-sequence-likelihood estimation. As a result, over 560000 note sequences were used for both $N = 2$ and $N = 3$ note-sequence models.

The note-sequence likelihoods are estimated by calculating the occurrences of different note sequences in the database. The procedure is demonstrated by the following example. Figure 4.11 shows an example melody consisting of MIDI notes 67, 63, 58, 55, 55, 51, 50, 53, 58, and 68 ($n_1 = 67, \ldots, n_{10} = 68$). The corresponding directed intervals between the notes, (-4, -5, -3, 0, -4, -1, $+3$, $+5$, and $+10$), have been marked above the score. The key of the melody is E♭ major, corresponding to the tonic note $k_{maj} = 3$. Given that $N = 3$, the first three-note sequence would be defined by a note-sequence-interval vector $[-4, -5]$ and a tonic distance $d_3 = \mathrm{mod}(67 - 3, 12) = 4$ in the major keys. Subsequentially, the second note sequence would be defined by an interval vector $[-5, -3]$ and a tonic distance $d_4 = \mathrm{mod}(63-3, 12) = 0$. Eventually, each note sequence in the database is classified in this manner to count the occurrences of different note sequences.

The estimation results for $N = 3$ are illustrated in Figures 4.12, 4.13, and 4.14 on the page 52, where note-sequence occurrences are represented for both major and minor keys with tonic-distance values $d = 0$, $d = 1$, and $d = 7$, respectively. For example, the Figure 4.12 reveals that there exists over 6000 occurrences of note sequences in the database for which the interval vector is $[4, -2]$, and the tonic distance is $d = 0$ for major key. Because the database contains more major-key melodies than minor-key melodies, there exists correspondingly less minor-key note-sequence occurrences.

The estimated note-sequence likelihoods may now be used to evaluate the occurrence likelihood of note sequences. For example, Figure 4.15 represents a part of a

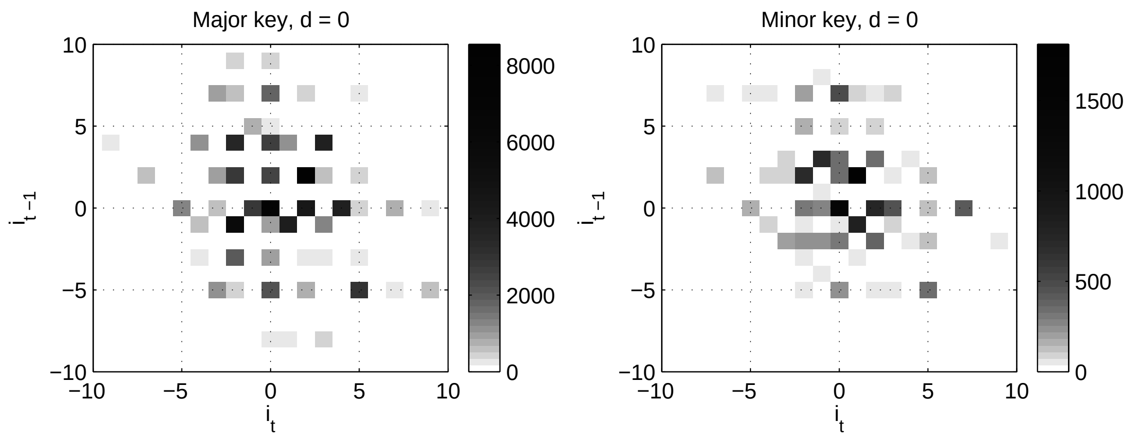

Figure 4.12: Note-sequence occurrences for $d = 0$.

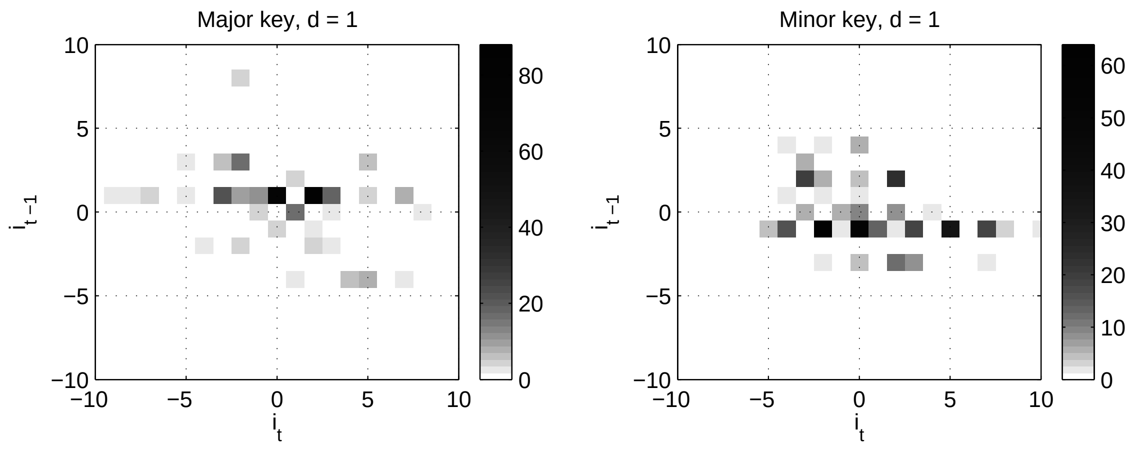

Figure 4.13: Note-sequence occurrences for $d = 1$.

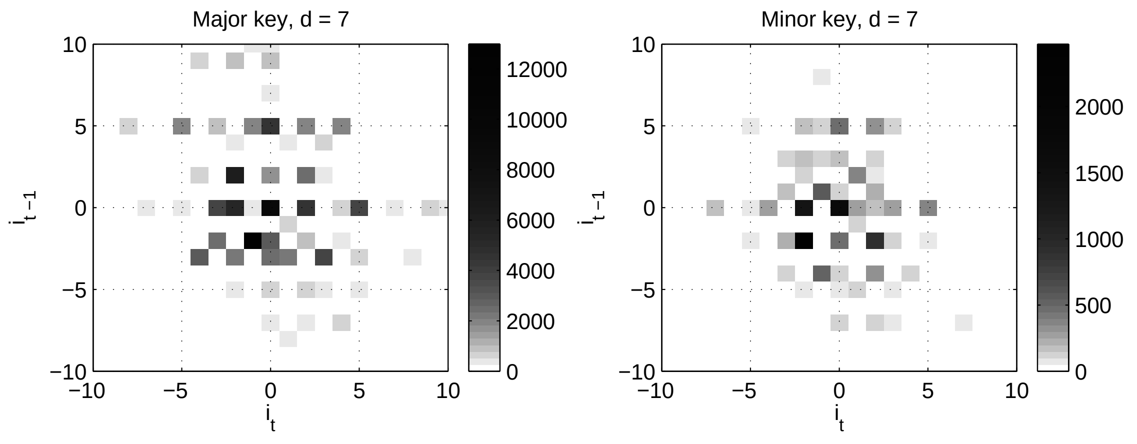

Figure 4.14: Note-sequence occurrences for $d = 7$.

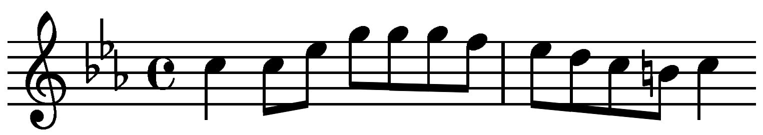

Figure 4.15: An excerpt of a Finnish folksong melody, *Taivas on sininen ja valkoinen*, in C minor.

traditional folksong in C minor key (i.e., $k_{min} = 0$), consisting of MIDI notes 60, 60, 63, 67, 67, 67, 65, 63, 62, 60, 59, and 60. Considering the last three-note sequence 60, 59, and 60 with a tonic distance $d = 0$, the interval vector $[-1, 1]$ has commonly occurred in the melody database according to the estimation results (Figure 4.12, minor key with $i_{t-1} = -1$, and $i_t = 1$). This is particularly interesting since MIDI note 59 does not even belong to the natural C minor scale[1].

In addition, the results can be used to *predict* the following note after a two-note sequence. For example, predicting the following note for a two-note sequence 67, 65 ($d = 7$, and $i_{t-1} = -2$), the note-sequence occurrences in Figure 4.14 suggest that i_t would be most likely -2, 0, or 2, thus producing the following-note candidates 63, 65, and 67. In fact, the example melody contains this type of a predictable note sequence with MIDI notes 67, 65, and 63.

Note N-grams

Although the estimated note-sequence likelihoods could be applied as such, the likelihoods are converted into note N-grams in order to combine the musicological model with the rest of the transcription system. The note N-grams are defined similarly to the conventional N-grams introduced in Section 4.1, thus stating the likelihood to move to a certain note given $N - 1$ previous notes. For $N \in \{2, 3\}$, the N-gram likelihood distribution A_{notes} is defined by Equations 4.23 (note *bigrams*) and 4.24 (note *trigrams*), respectively.

$$A_{notes} = \{a_{notes}(i, j)\}, \qquad a_{notes}(i, j) = P(n_t = j | n_{t-1} = i) \tag{4.23}$$

$$A_{notes} = \{a_{notes}(h, i, j)\}, \qquad a_{notes}(h, i, j) = P(n_t = j | n_{t-2} = h, n_{t-1} = i) \tag{4.24}$$

If some of the previous notes are unknown (in the beginning of melody), the first known note value is used instead of the unknown notes. For example, if note h is unknown for the note trigrams, $a_{notes}(i, i, j)$ is used. In addition, the influence of the note N-grams was adjusted by raising all the values $a_{notes}(i, j)$ and $a_{notes}(h, i, j)$ to a power of 10.

Despite the size of the database, there exists note sequences that do not occur in the database at all, forcing some of the distribution values to zero. However, melodies to be transcribed may contain such note sequences; as a consequence, the estimated likelihoods prevent those note sequences to be found. Therefore, the N-gram distribution values must be *smoothed* to address tiny likelihood values for the note sequences that were not found in the database.

[1]However, it does belong to the *harmonic* C minor scale.

The N-gram distribution smoothing is performed using the Witten-Bell discounting algorithm, originally proposed by Witten and Bell in [Witten91]. The algorithm is based on a principle of zero-frequency N-grams, i.e., the N-grams that have not yet occurred in the database. The zero-frequency N-gram likelihoods, however, may be estimated by the N-grams that have occurred *once*. Consequently, the zero-frequency likelihoods may be used to smooth the distribution. The Witten-Bell discounting algorithm is well known, and it is illustratively represented in [Jurafsky00], for example.

4.3.3 Using key estimation with note-sequence likelihoods

Sections 4.3.2 and 4.3.1 introduced the estimation of note-sequence likelihoods and musical keys, which are integrated in this section to constitute the musicological model of the transcription system. The musicological model is purposed to control the transitions between note events. In particular, this controlling is executed by using the N-gram likelihood distribution A_{notes} to determine the transition likelihoods between notes. However, the note-sequence likelihoods were defined by using the tonic distance d which is further affected by the tonic-note pitch classes k_{maj}, k_{min}. Therefore, the tonic notes must be determined by using the key estimation and accommodated in the A_{notes} distribution.

The key estimation is used with the note-sequence likelihoods as follows. First, use the key estimation to determine the tonic notes k_{maj}, k_{min} for the most probable relative-key pair. Second, choose the A_{notes} N-gram distributions for both major and minor keys corresponding to the tonic notes. Finally, sum the distributions together considering the relation between major and minor keys (Equation 4.15) and divide them by two, thus producing a mean N-gram distribution accommodating the relative major and minor keys found by the key estimation. However, if key estimation is disabled, all the note-sequence likelihoods are summed together and divided by 24 (the number of keys), thus ignoring the influence of key estimation.

4.4 The note model and the musicological model combined

Now that the note model and the musicological model have been proposed in sections 4.2 and 4.3, the models are combined to obtain a probabilistic note network. The network consists of note models through which the extracted musical features flow. Further, the musicological model controls the flow between the note models. Using this note network to transcribe melodies, the network assigns likelihoods for each path through the network. Consequently, the most probable path determines the optimal note sequence, thus producing a transcribed melody which is statistically the most likely according to the employed models.

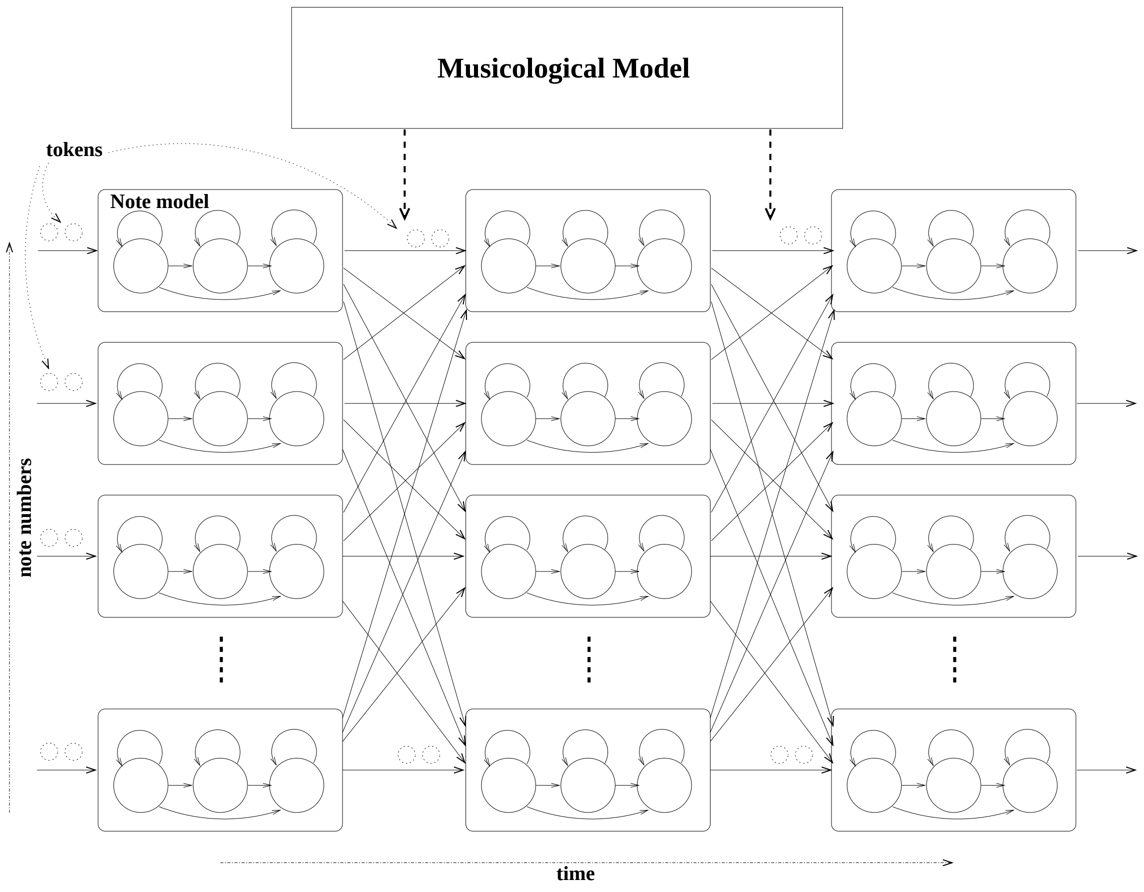

Figure 4.16: The combination of the note models and the musicological model.

4.4.1 Overview of the combined model

The combined model establishes a network structure consisting of note models and transitions between them. Each note model represents a note n with a specific MIDI note number, e.g., from MIDI note 36 to 96 that specifies the range of feasible note values in the transcription. Second, the transitions between the note models are controlled by the musicological model. In particular, the musicological model uses the note-transition likelihoods A_{notes} to control the transitions between the note models. The combination of the note models and the musicological model is illustrated in Figure 4.16. Notice that the figure represents the note models at every time instant even though there actually exist only one note model for each MIDI note (not for every time instant).

Accordingly, the combined model performs the following procedures for an observation vector $\mathbf{o}_t$ containing the musical features extracted at time t:

1. Update the musicological model to obtain A_{notes} (explained in Section 4.3.3).
2. Determine the observation likelihoods $b_{j,n}(\mathbf{o}_t)$ for each note model state $s_j \in S$ for each note model n (explained in Section 4.2.3).

3. Find the optimal path through the note network at time t (explained in Section 4.4.2).

In this way, the introduced note network may be used to transcribe monophonic melodies by finding the optimal note path through the network. This is explained in the following section.

4.4.2 Finding the optimal note sequence

The optimal path through the note network is determined by a so-called *token-passing algorithm*, proposed by Young *et al.* in [Young89]. The algorithm was originally designed for large-vocabulary speech-recognition systems and it is very similar to another extensively used method called *Viterbi algorithm* [Viterbi67, Forney73].

Token-passing algorithm represents a fundamental idea of *tokens* which consist of two items: a path identifier p, and a token weight w which specifies the weight of the path p through the connected network of HMMs. In general, tokens represent a flow through the network; essentially, the token with the smallest weight indicates the optimal path. The optimal path can be found by propagating tokens through the network, meaning that every state of a single HMM in the network emits tokens to the connected states in the model. Inside the HMMs, the token weights are increased by the observation likelihoods and the HMM transition probabilities. Furthermore, the token weights are also increased by the transition probabilities between the HMMs.

This idea of propagating tokens leads to an enormous amount of tokens in the network. For this reason, the token-passing algorithm applies a sub-optimality principle to reduce the number of tokens. The sub-optimality principle forces the algorithm to discard the tokens that seem to have too much weight. The principle is implemented so that each HMM state in the network takes in only a token with the smallest weight among the tokens emitted to the state. As a consequence, the token-passing algorithm just greedily preserves only the most probable paths at that time.

The token-passing algorithm solves the optimal path by using a structure called *Word Link Record* (WLR) which simply contains information about word boundaries[2], i.e., transitions between HMMs in the network. Path identifiers p are pointers to WLRs. Whenever a HMM emits a token out of the model at time t, a new WLR is created which contains the following items: the weight w and the path identifier p of the emitted token, the current time t, and the identity of the HMM that emitted the token. At this stage, there are two important things to do:

1. Append the created WLR to a list of WLRs.
2. Update the path identifier p of the emitted token to point to the created WLR *before* the token is propagated forward.

[2]Since the token-passing algorithm was originally designed for speech recognition, the network HMMs were modelling spoken words. In this thesis, a more appropriate term would be a *note boundary*.

As a consequence, the list of WLRs contains the information about the token propagation through the network. More importantly, the optimal path at time t can be found by searching the WLR with the smallest weight w and time value t. Since the WLR contains a pointer p to the preceding WLR (i.e., the preceding transition between HMMs), the optimal path is obtained by backtracking through the list. In particular, the WLR information specifies the HMM identities through which the path undergoes, and the difference between the time values of consecutive WLRs define the times that have been spent in each HMM.

Applying Token-passing algorithm to the note network

After introducing the fundamental idea of the token-passing algorithm, the algorithm is applied to the note network. Given the j:th state of the note model n, $s_{j,n}$, the algorithm steps are the following.

- Initialisation, $t = 0$:
  1. Initialise the first states of every note HMM (i.e., the states $s_{1,n}$) with tokens having weight $w = 0$.
  2. Initialise all the other states (i.e., the states $s_{j,n}$ for which $j \neq 1$), with tokens having weight $w = \infty$.
  3. Initialise the path identifiers of every token in the note model states to point to the beginning of the empty WLR list.
- Token passing, $t = 1, 2, \ldots, T$:
  1. **Inside the note models.** Pass a copy of the token in the state $s_{i,n}$ to states $s_{j,n}$ for which $a_{ij} \neq 0$, and increment the weight of the copied token with $b_{j,n}(\mathbf{o}_t) + a_{ij}$.
  2. **Between the note models.** First, get the tokens in the final states of the note models and add them to the WLR list. Second, copy these these tokens to the first states of all connected note models, and increment the token weights with the values in A_{notes}.
  3. In every note-model state in the network, find the token with the smallest weight, save it to the state, and discard all the other tokens emitted to the state.
- Termination, $t = T$:
  1. Search the WLR with the smallest weight and time value T from the list of WLRs.
  2. Follow the WLR list according to the path-identifier pointers to the beginning of the list to obtain the optimal path through the network.

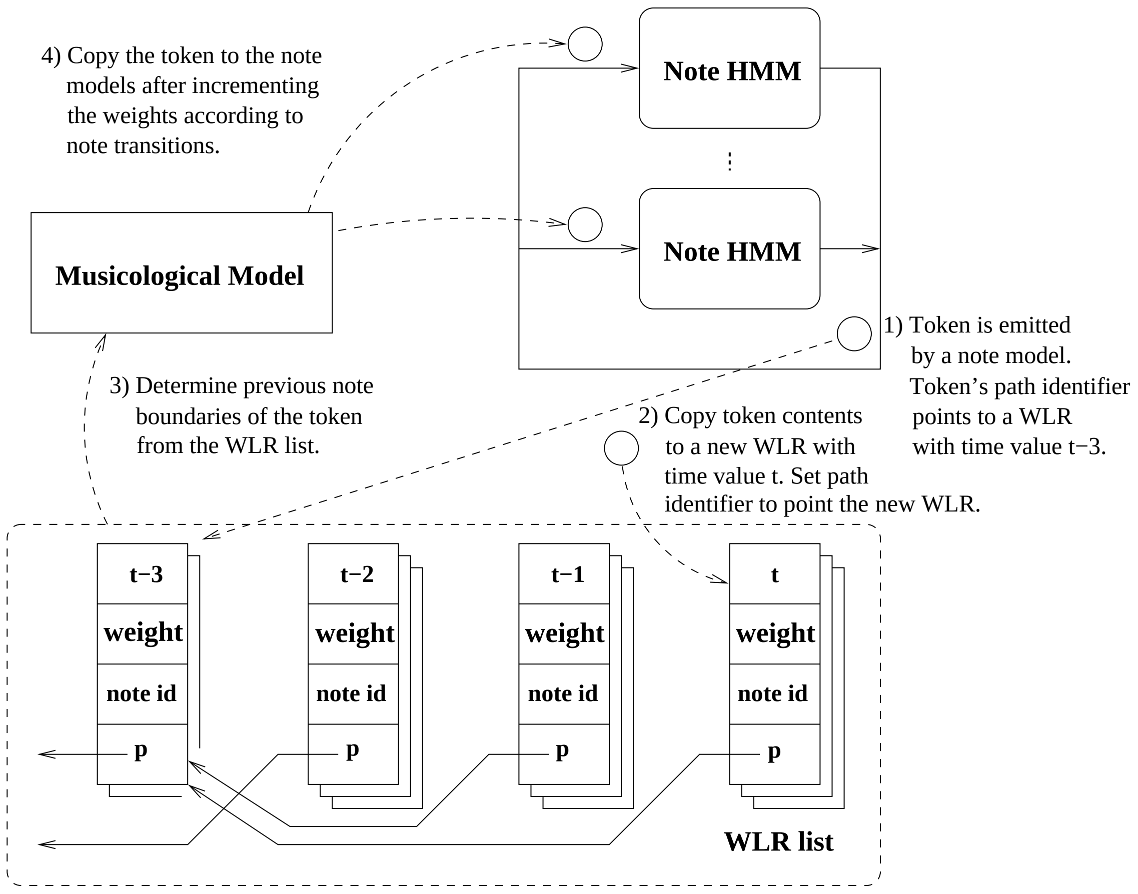
4) Copy the token to the note models after incrementing the weights according to note transitions.
Note HMM
Note HMM
Musicological Model
1) Token is emitted by a note model. Token's path identifier points to a WLR with time value t−3.
3) Determine previous note boundaries of the token from the WLR list.
2) Copy token contents to a new WLR with time value t. Set path identifier to point the new WLR.
t−3
t−2
t−1
t
weight
note id
p
WLR list

Figure 4.17: Using WLR list and note-transition likelihoods.

Figure 4.17 illustrates the use of the WLR list and the note-transition likelihoods A_{notes} to increment token weights between note models. First, a token is emitted by a note model at time t, and the path identifier of the token points to a WLR with time value $t-3$. Second, the contents of the token are copied to a new WLR, and the token's path identifier is set to point to the new WLR. Since the contents of the token were copied to the new WLR, the new WLR holds the path identifier that points to the WLR with time value $t-3$. Third, the previous note models that the token has visited can be determined by following the path identifier in the new WLR (e.g., the previous note of the token is indicated by the note identifier in the WLR with time value $t-3$). The previous note boundaries are passed to the musicological model which determines the note-transition likelihoods for the token to be passed to different note models. Finally, the token is copied to each note model after incrementing the token weight according to these note-transition likelihoods. Eventually, the optimal path is obtained by searching the WLR with smallest weight with last time value and following the path identifiers in this WLR to the beginning of the WLR list.

5 Melody Transcription System

In this chapter, the musical-feature extraction and the probabilistic models described in chapters 3 and 4 are integrated to constitute a melody transcription system. The system extracts note sequences from acoustic signals that contain monophonic melodies. In particular, the melodies are assumed to be represented by a human voice. However, the same architecture can be applied to transcribe monophonic melodies represented by any instrument. This requires that the note event model parameters are trained for the desired instrument, but otherwise the system remains unchanged. An overview of the system is illustrated in Figure 5.1, where a music signal is first processed to extract musical features, and then the features are interpreted by the probabilistic models to obtain a note sequence representing the performance.

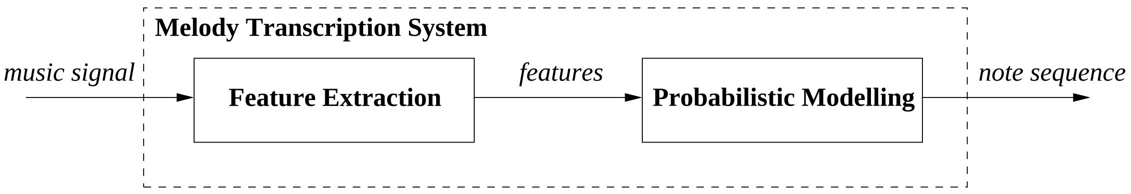

Figure 5.1: Melody transcription system.

This chapter is organised as follows. Section 5.1 presents the architecture of the melody-transcription system, and Section 5.2 shows some transcription examples. Section 5.3 introduces a real-time application for monophonic-melody transcription based on the proposed components.

5.1 System architecture

The transcription system consists of two main components which are the musical-feature extractors and the probabilistic models. Here, the musical-feature extractors include pitch, voicing, phenomenal accent, and metrical accent estimators. In addition, the extracted features may be post-processed in a desired way; in this work, pitch estimates are processed by the tuning algorithm. Second, the probabilistic models are applied to produce a meaningful interpretation of the extracted features, i.e., a note sequence.

The proposed melody-transcription system is represented in Figure 5.2, including all the components introduced in this thesis. First, the input signal is processed with

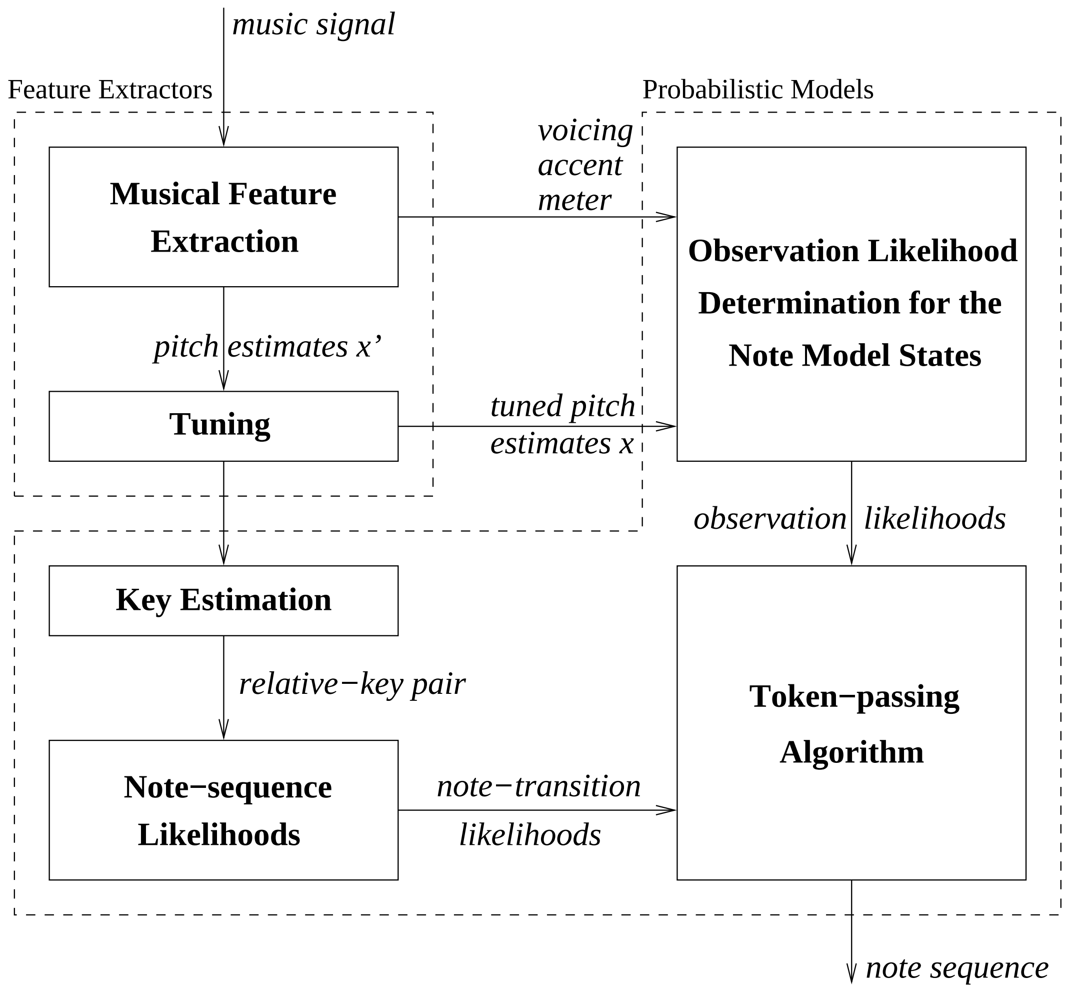

Figure 5.2: The architecture of the proposed melody-transcription system.

the feature extractors. Pitch estimates and the voicing of the frames are extracted with the YIN algorithm. In addition, the signal accent and meter are extracted by the algorithms discussed in Chapter 3. Further, the voiced pitch estimates are tuned and used with the other extracted features to constitute observation vectors, which are passed to the note-event models. The note-event models calculate state-conditional observation likelihoods for the features. At the same time, the tuned pitch estimates are delivered to the key-estimation model, where the most probable relative-key pair is determined and used to choose the note-transition likelihoods from the estimated note-sequence data. Finally, the observation likelihoods and the note-transition likelihoods are applied by the token-passing algorithm to produce a transcribed note sequence.

The melody-transcription-system architecture strives for the modularity of the components to enable a straightforward integration of new components into the system. For example, a new musical feature could be easily integrated into the system by training the note-event model with the new feature. In addition, the musicological model can be quite easily extended with new musicological assumptions and models due to the probabilistic note network used by the token-passing algorithm.

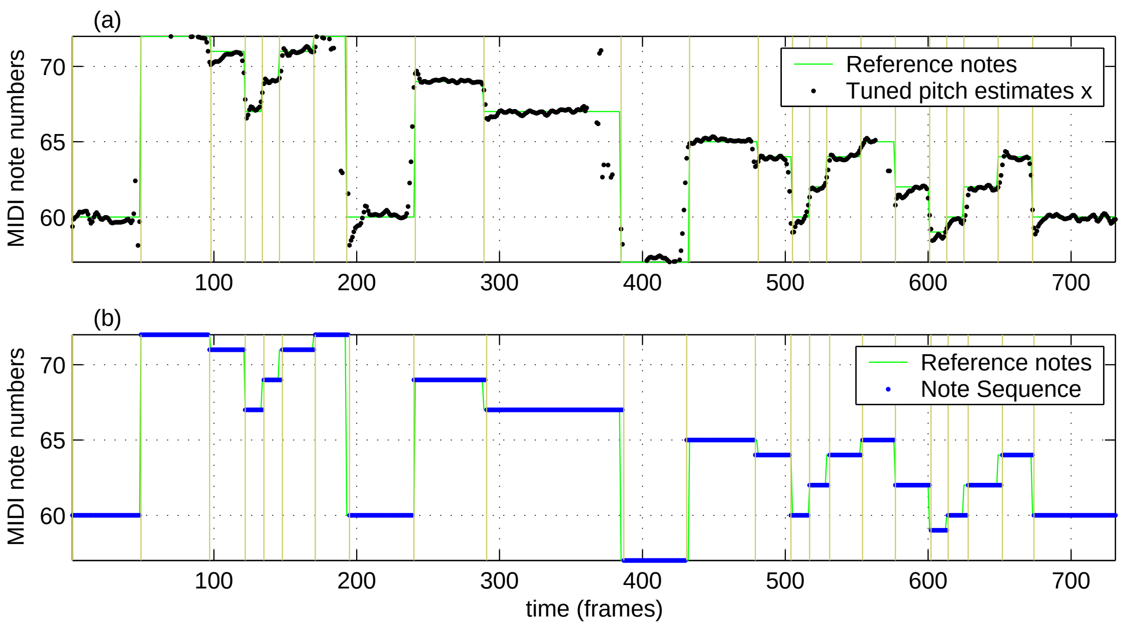

Figure 5.3: A simple transcription case.

5.2 Melody transcription examples

The operation of the system is here demonstrated with three practical melody-transcription cases: (i) a simple transcription case, (ii) a difficult transcription case, and (iii) a typical transcription case exemplifying the significance of the probabilistic models. All the cases are processed with the transcription system using a three-state note model with two GMM components for the observation likelihoods, note trigram models, and the feature set of tuned pitch estimates, voicing, and the accent signal.

The transcription cases are illustrated in figures 5.3-5.5 where the tuned pitch estimates x and the transcribed note sequence are displayed in panels (a) and (b), respectively. In addition, the reference melody is displayed in both (a) and (b) with a solid line. The vertical lines denote the beginnings of the reference-melody notes in the panels (a) and the beginnings of the transcribed-melody notes in the panels (b).

The simple transcription case in Figure 5.3 is performed by a female singer. The singer has performed the melody very accurately, since most of the frames are voiced, and the tuned pitch estimates follow the reference melody closely as shown in panel (a). Consequently, the correct note sequence can be easily inferred by the transcription system as shown in panel (b).

The second transcription case is much more difficult than the first, and it is illustrated in Figure 5.4. The melody performed by a male singer contains several inaccuracies. For example in the beginning of the melody, the performer strives for a MIDI note 57 but catches the note by gliding to it only after one second. This gliding is interpreted by the transcription system as a rapid series of notes which is called a *glissando* in music terminology. The notes defining the glissando are not

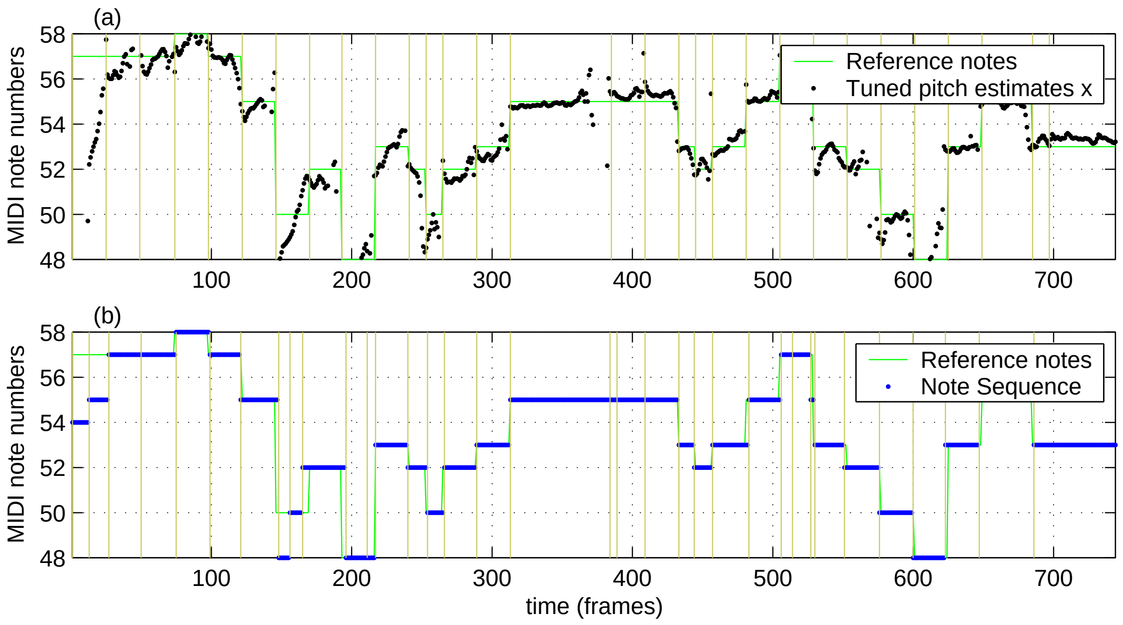

Figure 5.4: A difficult transcription case.

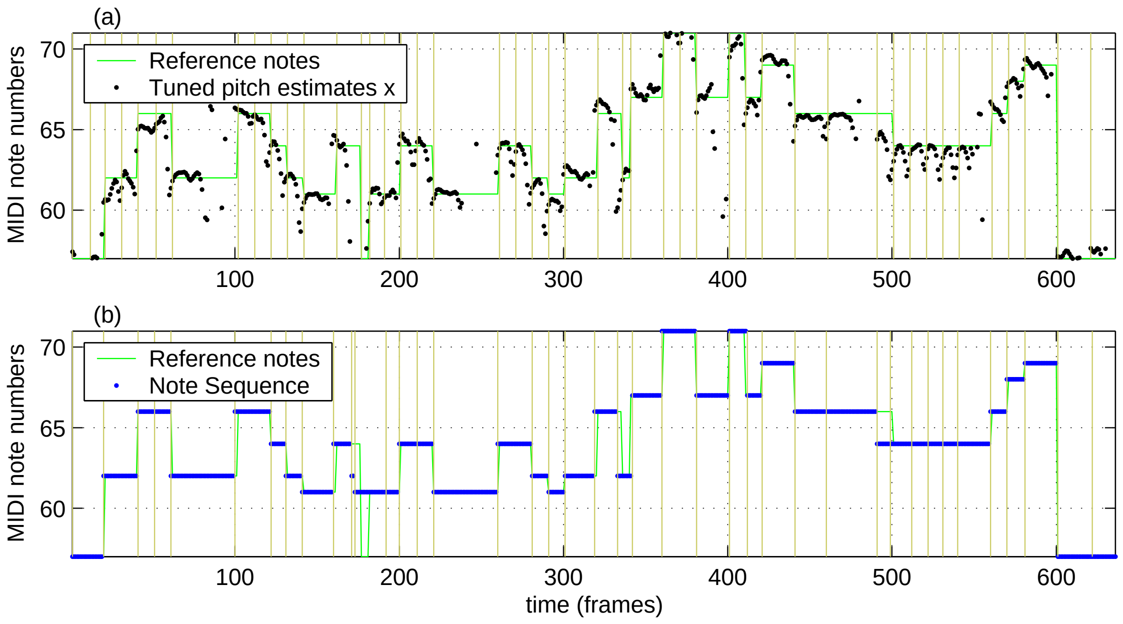

Figure 5.5: A typical transcription case.

only determined by the pitch estimates but also by the musicological model which suggests a sequence of notes 54, 55, and 57. The MIDI note 54 is exceptional in the F major key, which is nonetheless correctly estimated by the key-estimation model. Consequently, there should not exist MIDI note 54 in the glissando. This can, however, be explained by the fact that the glissando occurs in the beginning of the melody, and there exists no preceding notes that could be used to determine the first note value. On the other hand, all the other notes in the transcribed note sequence belong to the F major key, and the transcription result is decent.

The final transcription case represents a normal degree of difficulty in melody transcription, and it is shown in Figure 5.5. The melody performed by a female singer is in D major key, for which the key has been correctly estimated. This case expresses the significance of the probabilistic models; first, the musicological model corrects the third note of the transcribed note sequence to a MIDI note 66, although the pitch estimates are more close to a MIDI note 65, for example. Second, the singer produces rapid slides to notes, particularly in the long sequence of a MIDI note 64 at the end of the melody. These slides are handled by the note model that considers them as a part of the note events, thus producing a correctly transcribed note sequence.

In general, the melody-transcription system follows the performer quite faithfully, and the probabilistic models produce their contribution only after the performance contains ambiguity or inaccuracies. In such cases, the system attempts to produce a transcription that *sounds* good in a common way. However, the transcription system can also transcribe unconventional note sequences, if the performance is just accurate enough.

5.3 Real-time melody transcription application

The proposed algorithms of the melody-transcription system were also integrated with an application which was designed for testing transcription algorithms in real-time situations. The application was implemented in C++, and it shows a real-time visualisation of the transcription while the input signal is received from the computer sound card. The most important features of the application are the following:

- A graphical user interface implemented with the *wxWindows* library which is a cross-platform framework for graphical user-interface development in C++. For more information on wxWindows, please visit `www.wxwindows.org`.
- Real-time processing of the input signal from the computer sound card. The transcription results are immediately displayed on the screen.
- Conversion of the transcription results into the Standard MIDI File 1.0 format [MIDI96]. In addition, MIDI files can be imported to the program and displayed on the screen.
- An interface for adjusting the parameters of the transcription algorithm.

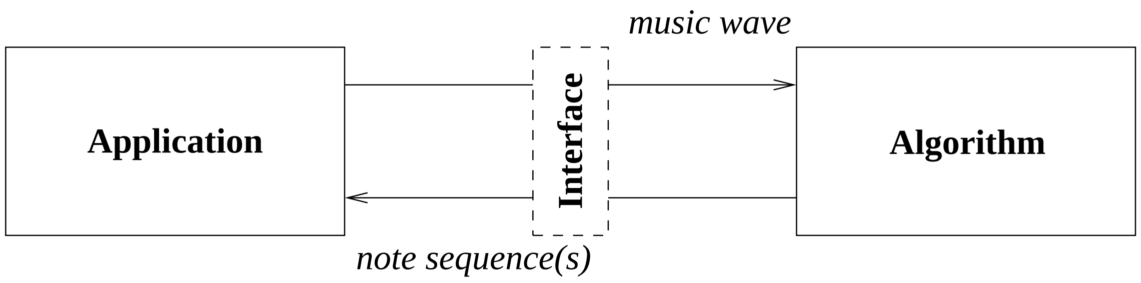

Figure 5.6: The generic music-transcriber interface.

The application design also defines a generic interface for transcription algorithms. The interface requires that a transcription algorithm takes an acoustic time-domain signal as its input and produces musical notes in MIDI format as the output. Consequently, any type of transcription algorithm can be integrated with the application. In addition, several algorithms can be used in parallel. This provides a convenient way for comparing different transcription algorithms. Figure 5.6 shows the interface between the application and a transcription algorithm. The application manages the communication with the computer sound card, the visualisation of transcribed note sequences, and the interaction with the user, whereas the algorithm itself merely produces a note sequence corresponding to the sound signal received from the application.

Figure 5.7 shows the main window of the application. The application controls and menus are placed on the top of the window, and the MIDI-note grid is indicated by the piano keys on the left. The transcription results are displayed at the centre of the screen. The time line and the application information field are positioned at the bottom of the window. In addition, there exist a signal-level meter at the left upper corner of the window to monitor the peak level of the input signal.

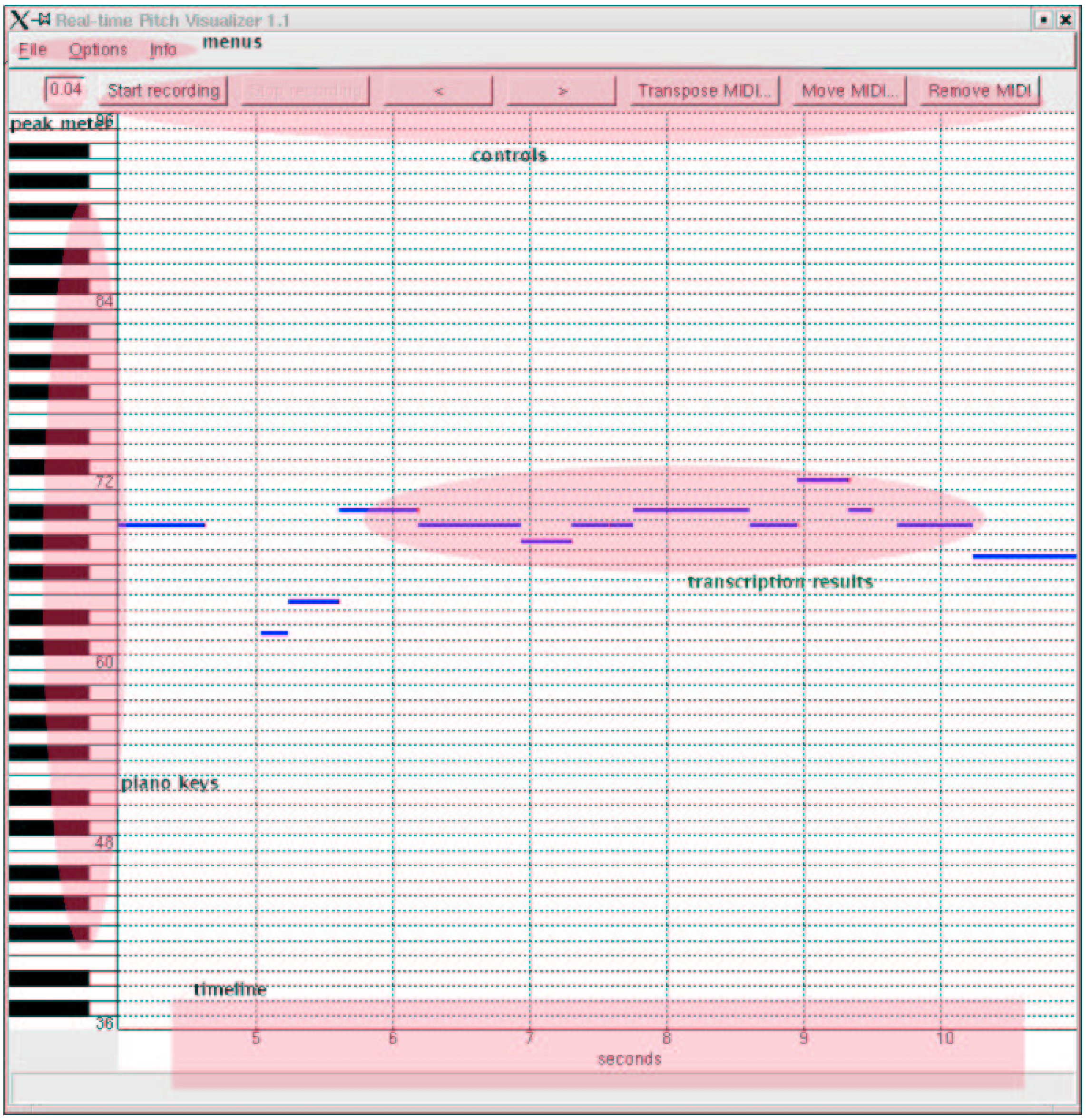

Figure 5.7: The main window of the melody-transcription application.

6 Simulation Results

This chapter presents the evaluation of the proposed transcription system. The evaluation is performed by simulating melody-transcription situations with different system setups. The correctness of the transcription is quantitatively evaluated and the results are reported. Section 6.1 represents the simulation setups and Section 6.2 explains the criteria of evaluation. Finally in Section 6.3, the evaluation results are reported and discussed.

6.1 Simulation setups

The proposed melody-transcription system is evaluated by using the same acoustic database that was used to train the note models. In addition to the performed melodies, the database contains the corresponding reference melodies, thus enabling the comparison of the transcription results with the reference melodies. The evaluation uses 57 melodies from the database that are performed by two male and two female non-professional singers. The melodies used in evaluation contain over 2200 notes in total.

The simulation setups are defined by the following parameters: (i) the number of note-model states K, (ii) the number of GMM components η for a note-model observation distributions, (iii) the used musical-feature set for the note model, (iv) the note N-gram size, (v) enabling of the key estimation, and (vi) enabling the use of the musicological model. Particularly, if the musicological model is not used, we use an equal transition likelihood for all the transitions between notes. The simulation-setup parameter values are represented in Table 6.1. Altogether, the parameter combinations result in 600 simulation setups. The weighting of the musical-feature

Parameter	Values
number of note-model states, K	1, 2, 3, 4, 5
number of GMM components, η	1, 2, 3, 4, 5, 6
feature set	$\{x_\Delta, \nu\}, \{x_\Delta, \nu, a\}, \{x_\Delta, \nu, m\}, \{x_\Delta, \nu, a, m\}$
note N-gram size, N	2, 3
key estimation	enabled, not enabled
musicological model used	yes, no

Table 6.1: The simulation-setup parameters.

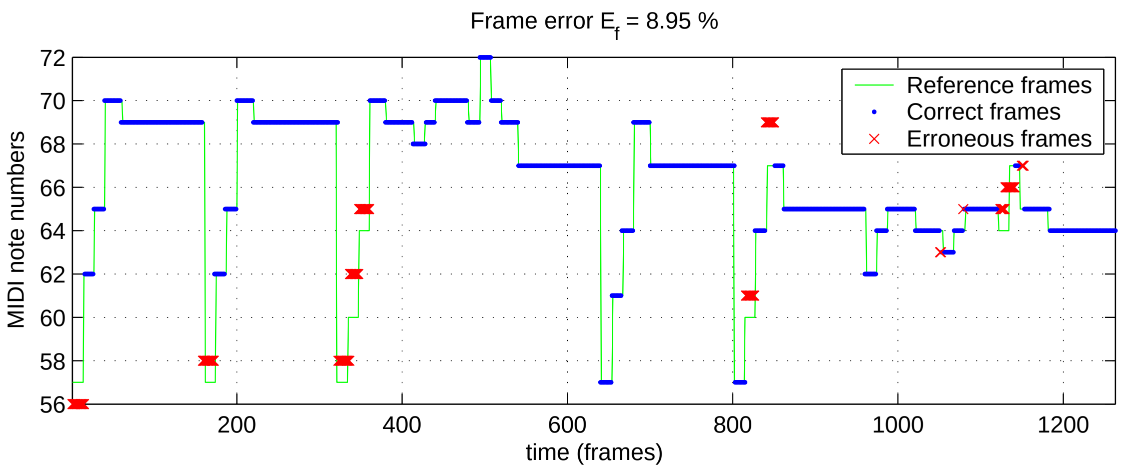

Figure 6.1: The frame-based evaluation of a transcribed melody.

sets and the note-transition likelihoods also affect the simulation results. However, these weightings are empirically determined and fixed already at an earlier stage as explained in Section 4.2.3. To have a rough baseline reference system for the proposed system, a simple rounding of pitch estimates to the nearest MIDI notes is performed and evaluated.

6.2 Melody transcription evaluation

In general, melody-transcription systems can be qualitatively or quantitatively evaluated. In this work, we employ quantitative evaluation by measuring the difference between the reference melodies and the transcribed melodies. Particularly, the difference is examined with two evaluation criteria: a *frame-based* criterion, and a *note-based* criterion. These are explained in the following sections.

6.2.1 Frame-based evaluation

The frame-based evaluation criterion is defined by the number of correctly transcribed frames c_{cor} and the number of voiced frames c_{ref} in the reference melody. First, a frame is considered to be correctly transcribed, if the transcribed note equals to the reference note in that frame. However, the evaluation database contains performances that are slightly unsynchronised in time compared to the reference melodies, thus producing an inessential growth of erroneously transcribed frames. Therefore, two note values at the reference-note boundaries are considered to be correct within a $\pm$50 ms distance from the note boundary (i.e., $\pm$2 frames). Second, the voiced frames mean that the reference melody contains a note-pitch value in that frame, meaning that note rests in the reference melodies are not included in the evaluation. However, the database contains only few of them, thus the effect for the results is insignificant.

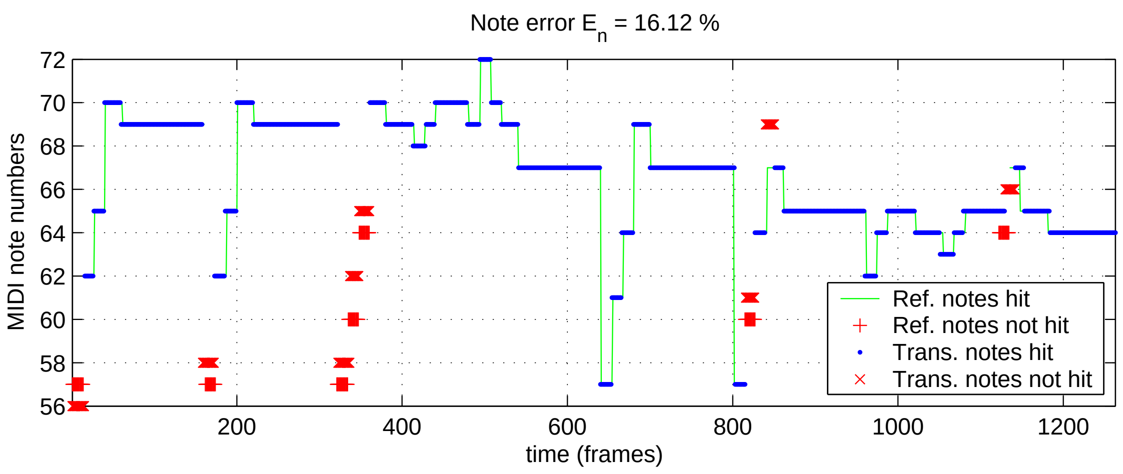

Figure 6.2: The note-based evaluation of a transcribed melody.

The *frame error* E_f for a transcribed melody is defined as

$$E_f = \frac{c_{ref} - c_{cor}}{c_{ref}} \cdot 100\%. \tag{6.1}$$

As an example, the frame-error is evaluated for a transcribed melody in Figure 6.1. The frame errors are calculated for each individual melody in the evaluation melodies database and the average of these errors is reported.

6.2.2 Note-based evaluation

In contrast to the frame-based evaluation, the note-based evaluation criterion uses notes rather than frames as the evaluation units. Considering notes as meaningful units, the faults of the frame-based evaluation, such as the influence of note length, can be avoided. Moreover, the significance of notes should be emphasised, since we are evaluating the correctness of note sequences.

The note-based evaluation is symmetrically approached from both the reference and the transcribed melodies point of view. First, we count the number of reference notes that are *hit* by the transcribed melody and denote this number with $\check{c}_R$. A reference note is hit, if a note in the transcribed melody overlaps with the reference note both in time and in note value. Second, the same scheme is applied so that the reference and transcribed melody exchange roles, i.e., we count the number of transcribed notes that are hit by the reference melody and denote the count with $\check{c}_T$.

The *note error* E_n for a transcribed melody is defined as

$$E_n = \frac{1}{2}\left[\frac{c_R - \check{c}_R}{c_R} + \frac{c_T - \check{c}_T}{c_T}\right] \cdot 100\%, \tag{6.2}$$

where c_R is the number of reference notes, and c_T is the number of transcribed notes. Particularly, Equation 6.2 defines the average of the differences between the reference melody and the transcribed melody. This symmetrical approach is essential, because the use of either $\check{c}_R$ or $\check{c}_T$ only would lead to an evaluation criterion that could be eluded by the transcription system. If only the $\check{c}_R$ would be used, a zero note error could be produced by randomly transcribing notes to surely hit every note in the reference melody. On the other hand, if only the $\check{c}_T$ would be used, a zero note error could be attained by transcribing a single note that would surely be in the reference melody.

Similarly to the frame errors, the note errors are calculated for each melody in the evaluation database and the average of these is reported. As an example, the note-based evaluation of a transcribed melody is illustrated in Figure 6.2.

6.3 Results

The melody-transcription system achieved error percentages smaller than 10 percent with both evaluation criteria. The best results for different feature sets are represented in Table 6.2. By using a simple rounding of pitch estimates to the nearest MIDI notes, the corresponding frame error in the database was 20.3 %, whereas the feature set with pitch difference and voicing $\{x_\Delta, \nu\}$ achieved error percentages slightly over 10 %. When the accent feature a was included in the note model, error percentages decreased by approximately one percentage unit for both the frame and the note error criterion. Further, replacing the accent feature with the meter feature m reduced note errors and achieved the best performance according to the note error criterion; however, frame errors were increased. On the other hand, the fourth feature set including all the proposed features reached the best performance according to the frame-error criterion. All the best-performance setups used key estimation and note N-grams of length 2. Detailed simulation results are represented in Appendix A for all the feature sets, excluding the set $\{x_\Delta, \nu, a\}$ which is discussed in the following.

Table 6.3 shows the error percentages when using the feature set $\{x_\Delta, \nu, a\}$, key estimation, and note bigrams, i.e., note-sequence likelihoods with sequence length of 2. The table shows clearly the influence of the number of note-model states K and the number of GMM components η. The first column of the table shows the cases when only one GMM component ($\eta = 1$) is used and the number of states is varied; the error percentages decrease when the number of states in the note model is increased. Similar trend can be observed by fixing the number of states to 1 and

Feature set	E_f	E_n
$\{x_\Delta, \nu\}$	10.3	10.3
$\{x_\Delta, \nu, a\}$	9.2	9.6
$\{x_\Delta, \nu, m\}$	10.3	**9.4**
$\{x_\Delta, \nu, a, m\}$	**9.1**	9.9

Table 6.2: The best results of each note-model feature set.

K \ η	1	2	3	4	5	6	criterion
1	90.4	15.2	16.2	15.9	15.9	15.6	E_f
	64	19.5	18.4	18.6	18.3	18.2	E_n
2	72.8	12	13.9	13.2	13.1	13.2	E_f
	61.2	15.2	16.9	16	16.1	16.1	E_n
3	17.9	**9.2**	9.7	9.9	9.8	10	E_f
	18.1	**10.2**	10.6	10.4	10	10.2	E_n
4	18.8	9.4	9.8	10.1	**9.8**	10.2	E_f
	18.8	10.3	9.8	10	**9.6**	9.9	E_n
5	16.6	10.3	10.5	10.1	10.1	10.4	E_f
	15.3	11.2	11.2	10.4	10.4	10.4	E_n

Table 6.3: Error rates for the feature set $\{x_\Delta, \nu, a\}$ using key estimation and note bigrams.

varying the number of components. However, the increase of the number of states and components does not measurably improve transcription results after using three or four states in the note model and two GMM components. Moreover, the increase of components reduces the generality of the note model, and the increase of note model states does not improve the temporal-separation accuracy of the note. In general, the best results were achieved by using three or four states and two to five GMM components in the observation-likelihood distributions. As an exception, a five-state model with five GMM components performed best for the feature set $\{x_\Delta, \nu\}$.

Influence of the musicological model

The musicological model uses note N-grams and key estimation to determine transition likelihoods between notes. If the musicological model is not used, equal transition likelihoods for every note transition are used. Figure 6.3 shows the best results for each feature set when the musicological model is not used, key estimation is either enabled (ON) or not enabled (OFF), and note N-gram length N is either 2 or 3. Surprisingly by disabling the musicological model, the system performs approximately as well as using note bigrams ($N = 2$) without key estimation. This means that the good performance of the system is mostly a consequence of using the note-event model. However, using key estimation in the musicological model clearly improves the system performance. It was also unexpected that the use of note bigrams with key estimation produces a slightly better performance than using note trigrams. An explanation for this could be that the trained note trigrams are too specific to the training material (which differs from the test material) when trigrams are used with key estimation. However, if key estimation is disabled, note trigrams perform better since they are more general in that case (i.e., the distance from tonic note is ignored) and guarantee the musicality of three-note sequences

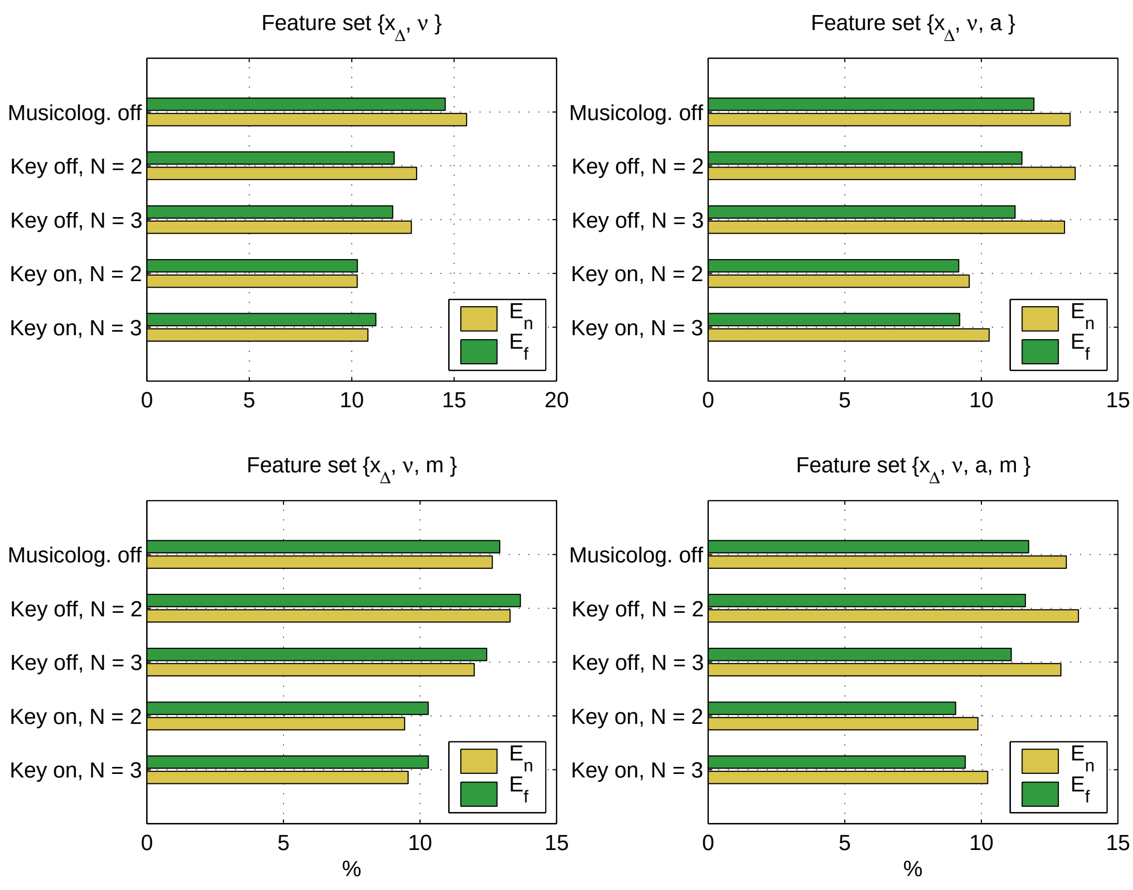

Figure 6.3: The influence of the musicological model and its parameters to the best results of each feature set.

instead of two-note sequences, although almost the same performance is achieved by completely disabling the musicological model.

Influence of the transcription system

A comparison between the transcription system output and the pitch estimates clarifies the influence of different parts of the system to the simulation results. Figure 6.4(a) shows a histogram of the differences between pitch estimates x and the reference notes in the database. The pitch estimates are produced by the YIN algorithm, which incidentally makes too-high octave errors. This can be observed twelve semitones above the reference notes. Rounding the pitch estimates to the nearest MIDI note numbers shows that the most of the errors occur one semitone below the reference notes, whereas the errors are more infrequent one semitone above the reference notes. This is clearly seen in the histogram in Figure 6.4(b). After using the probabilistic models, the amount of the semitone too-low and too-high errors have been at least halved as shown in Figure 6.4(c). The transcribed notes were produced by using a three-state note model with feature set $\{x_\Delta, \nu, a, m\}$, two GMM components for the observation distributions, key estimation, and note bigrams. However, the

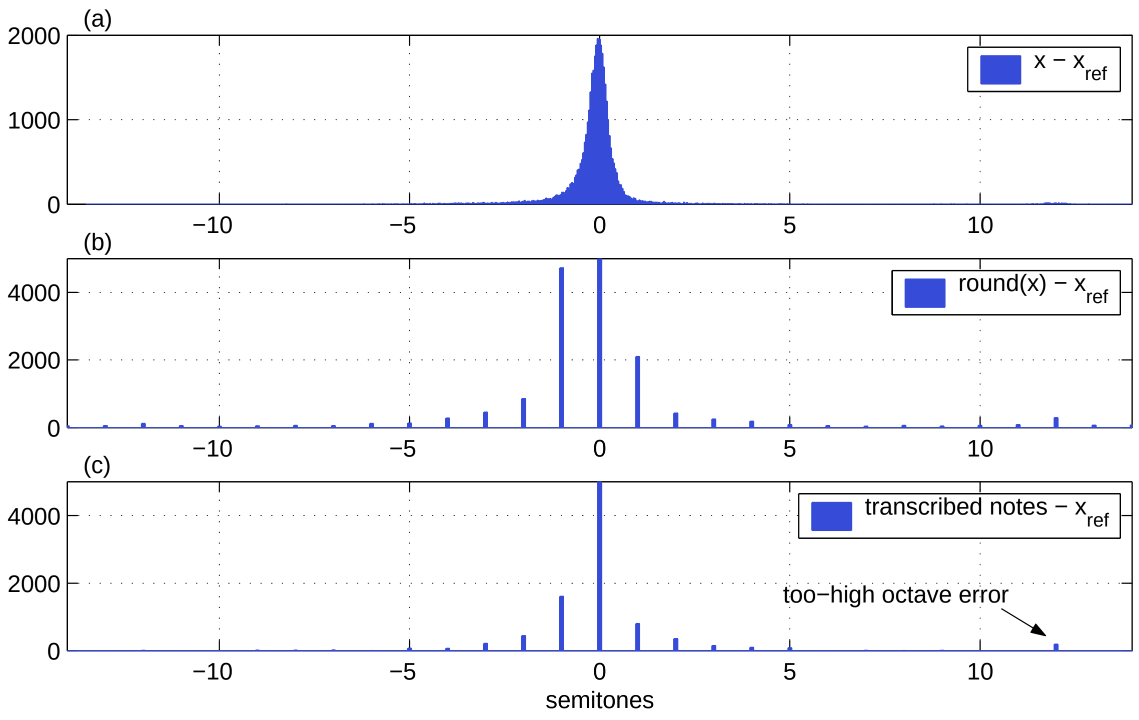

Figure 6.4: Errors in semitones.

transcription system fails to handle too-high octave errors produced by the YIN algorithm. Since the note model does not consider any octave errors, the optimal note sequence is obtained by visiting the octave too-high note despite the small likelihood to move to a such note according to the musicological model.

In general, the proposed transcription system transcribes monophonic vocal performances rather well. However, transcribed note sequences could be further postprocessed to attain even better-sounding transcriptions. For example, the durations of transcribed notes are not quantised or processed that would be required to construct a score notation of the performance. In addition, transcribed note sequences could be handled to satisfy general musical conventions (for example, similarity of note patterns in melodies). This type of approach could be based on the generative theory of tonal music examined by Lerdahl and Jackendoff in [Lerdahl83], for example.

7 Conclusions

In this thesis, we have investigated the problem of automatic transcription of monophonic melodies and proposed probabilistic models to solve it. We studied the perception of music signals and discussed the musical features that describe musically meaningful characteristics of melodies. The primary goal of the thesis was to apply methods used in speech recognition to the modelling of note events and to combine musicological knowledge with signal processing to obtain a melody transcription system.

The main components of the system were the note event model and the musicological model. The note model represented note candidates in melodies, particularly emphasising the importance of notes as the basic musical units with dynamic nature. The use of the model also enabled a more appropriate examination of the relationships of notes with each other by the musicological model. The note event was described with a hidden Markov model which used four musical features (pitch, voicing, accent, and metrical accent) to determine the likelihoods of different note candidates. Furthermore, the transition probabilities between note candidates were obtained using the musicological model which exploited key estimation and the likelihoods of two-note and three-note sequences. These two probabilistic models were integrated to form the transcription system having a modular architecture and, thus, to enable a straightforward integration of desired musical-feature extractors and musicological rules to the system.

In general, the proposed system achieved a good accuracy in transcribing monophonic melodies: over 90 % of the notes in the acoustic database were correctly transcribed, thus halving the amount of errors compared to the simple rounding of pitch estimates to the nearest MIDI note. The improvement is mostly due to the use of the note event model. The musicological model improved the results only when key estimation was enabled.

This work has described a transcription system that can be easily extended in the future. In particular, the system could be extended to transcribe polyphonic music by allowing melody lines and chords to be handled simultaneously. In addition, the integration of melody and rhythm transcription should be considered. In such a case, musicological modelling would become more important and the complexity of the system would inevitably increase. Nevertheless, these future directions open interesting topics of research, pursuing for the ultimate challenge of transcribing music on commercial compact discs.

Bibliography

[ANSI94] ANSI. ANSI S1.1-1994: American national standard acoustical terminology. Acoustical Society Of America, New York, 1994.

[Bregman90] A. S. Bregman. Auditory scene analysis: The perceptual organization of sound. MIT Press, 1990.

[Burns99] E. M. Burns. *Intervals, Scales, and Tuning.* In D. Deutsch, editor, The Psychology of Music, chapter 7, pages 215–264. Academic Press, second edition, 1999.

[Cheveigné99] A. de Cheveigné and H. Kawahara. *Multiple period estimation and pitch perception model.* In Speech Communications 27, pages 175–185. Elsevier Science B.V., 1999.

[Cheveigné02] A. de Cheveigné and H. Kawahara. *YIN, a fundamental frequency estimator for speech and music.* J. Acoust. Soc. Am., vol. 111, no. 4, pages 1917–1930, April 2002.

[Clarisse02] L. P. Clarisse, J. P. Martens, M. Lesaffre, B. De Baets, H. De Meyer and M. Leman. *An Auditory Model Based Transcriber of Singing Sequences.* In Proc. of 3rd International Conference on Music Information Retrieval, ISMIR '02, 2002.

[Clarke99] E. F. Clarke. *Rhythm And Timing In Music.* In D. Deutsch, editor, The Psychology of Music, chapter 13, pages 473–500. Academic Press, second edition, 1999.

[Dahlig94] E. Dahlig. *EsAC database: Essen Associative Code and Folksong Database.* `www.esac-data.org`, 1994.

[Davy03] M. Davy and S. J. Godsill. *Bayesian Harmonic Models for Musical Signal Analysis.* In J. M. Bernardo, J. O. Berger and Da, editors, Bayesian Statistics VII. Oxford University Press, 2003.

[Duda01] R. O. Duda, P. E. Hart and D. G. Stork. Pattern classification. John Wiley & Sons, Inc., second edition, 2001.

[Forney73] G. D. Forney. *The Viterbi Algorithm.* In Proceedings of the IEEE, vol. 61, pages 268–278, March 1973.

[Gómez03] E. Gómez, A. Klapuri and B. Meudic. *Melody description and extraction in the context of music content processing.* J. of New Music Research, 2003.

[Goldstein73] J. L. Goldstein. *An optimal processor theory for the central formation of the pitch of complex tones.* J. Acoust. Soc. Am., vol. 54, no. 6, pages 1496–1516, December 1973.

[Goto98] M. Goto and Y. Muraoka. *An Audio-based Real-time Beat Tracking System and Its Applications.* In Proceedings of International Computer Music Conference, ICMC, 1998.

[Gouyon01] 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, 2001.

[Handel95] S. Handel. *Timbre Perception and Auditory Object Identification.* In B. C. J. Moore, editor, Hearing, chapter 12, pages 425–461. Academic Press, Inc., second edition, 1995.

[Hartmann96] W. M. Hartmann. *Pitch, periodicity, and auditory organization.* J. Acoust. Soc. Am., vol. 100, no. 6, pages 3491–3502, December 1996.

[Heittola03] T. Heittola. Automatic classification of music signals. Master's thesis, Tampere University of Technology, December 2003.

[Jelinek97] F. Jelinek. Statistical methods for speech recognition. The MIT Press, 1997.

[Jurafsky00] D. Jurafsky and J. H. Martin. Speech and language processing. Prentice-Hall, Inc., 2000.

[Kashino95] K. Kashino, K. Nakadai, T. Kinoshita and H. Tanaka. *Organization of Hierarchial Perceptual Sounds: Music Scene Analysis with Autonomous Processing Modules and a Quantitative Information Integration Mechanism.* In Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), vol. 1, pages 158–164, August 1995.

[Klapuri00] A. Klapuri. *Qualitative and quantitative aspects in the design of periodicity estimation algorithms.* In Proceedings of the European Signal Processing Conference EUSIPCO, 2000.

[Klapuri03a] A. Klapuri. *Automatic Transcription of Music.* In Proceedings of the Stockholm Music Acoustics Conference SMAC 03, August 2003.

[Klapuri03b] A. Klapuri. *Musical Meter Estimation and Music Transcription.* In Proceedings of Cambridge Music Processing Colloquium. Cambridge Music Processing Colloquium, March 2003.

[Krumhansl90] C. Krumhansl. Cognitive foundations of musical pitch. Oxford University Press, first edition, 1990.

[Lerdahl83] F. Lerdahl and R. Jackendoff. A generative theory of tonal music. The MIT Press, 1983.

[Martin96] K. D. Martin. *A Blackboard System for Automatic Transcription of Simple Polyphonic Music.* Technical Report 385, Massachusetts Institute of Technology Media Laboratory Perceptual Computing Section, July 1996.

[Meddis91] R. Meddis and M. J. Hewitt. *Virtual Pitch and Phase Sensitivity of a Computer Model of the Auditory Periphery. I. Pitch Identification.* J. Acoust. Soc. Am., vol. 89, no. 6, pages 2866–2882, June 1991.

[Meddis97] R. Meddis and L. O'Mard. *A unitary model of pitch perception.* J. Acoust. Soc. Am., vol. 102, no. 3, pages 1811–1820, September 1997.

[MIDI96] The MIDI Manufacturers Association. *The Complete MIDI 1.0 Detailed Specification*, second edition, 1996. Website: `www.midi.org`.

[MPE98] *Information Technology - Coding of Audiovisual Objects - Low Bitrate Coding of Multimedia Objects.* MPEG-4 Standard, Final Committee Draft, 1998. Part 3: Audio, Subpart 5: Structured Audio. Available at `http://web.media.mit.edu/~eds/mpeg4`.

[Papoulis84] A. Papoulis. *Brownian Movement and Markoff Processes.* In Probability, Random Variables, and Stochastic Processes, chapter 15, pages 515–553. New York: McGraw-Hill, second edition, 1984.

[Paulus03a] J. Paulus and A. Klapuri. *Model-based event labeling in the transcription of percussive audio signals.* In Proceedings of the 6th Int. Conference on Digital Audio Effects (DAFX-03), September 2003.

[Paulus03b] J. K. Paulus and A. P. Klapuri. *Conventional and periodic N-grams in the transcription of drum sequences.* In Proceedings of International Conference on Multimedia and Expo (ICME 2003), 2003.

[Piszczalski79] M. Piszczalski and B. A. Galler. *Predicting musical pitch from component frequency ratios.* J. Acoust. Soc. Am., vol. 66, no. 3, pages 710–720, September 1979.

[Pollastri02] E. Pollastri. *A Pitch Tracking System Dedicated to Process Singing Voice for Musical Retrieval.* In 2002 IEEE International Conference on Multimedia and Expo, ICME '02, vol. 1, pages 341–344, August 2002.

[Rabiner76] L. R. Rabiner, M. J. Cheng, A. E. Rosenberg and C. A. McGonecal. *A Comparative Performance Study of Several Pitch Detection Algorithms.* IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 24, no. 5, pages 399–418, October 1976.

[Rabiner89] L. R. Rabiner. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* Proceedings of the IEEE, vol. 77, no. 2, pages 257–289, February 1989.

[Rowe01] R. Rowe. Machine musicianship. The MIT Press, 2001.

[Scheirer98] E. D. Scheirer. *Tempo and beat analysis of acoustic musical signals.* J. Acoust. Soc. Am., vol. 103, pages 588–601, January 1998.

[Scheirer00] E. D. Scheirer. *Music-Listening Systems.* PhD thesis, School of Architecture and Planning, MIT Media Laboratory, 2000.

[Selfridge-Field98] E. Selfridge-Field. *Conceptual and representational issues in melodic comparison.* Computing in Musicology, vol. 11, pages 3–64, 1998.

[Shih03a] H. Shih, S. S. Narayanan and C.-C. J. Kuo. *A statistical multidimensional humming transcription using phone level hidden markov models for query by humming systems.* Proceedings of IEEE 2003 International Conference on Multimedia and Expo, vol. 1, pages 61–64, 2003.

[Shih03b] H. Shih, S. Narayanan and C.-C. J. Kuo. *Multidimensional humming transcription using a statistical approach for query by humming systems.* In 2003 International Conference on Multimedia and Expo, ICME '03, vol. 3, pages 385–388, July 2003.

[Stevens75] S. S. Stevens. Psychophysics. Wiley & Sons, New York., 1975.

[Talkin95] D. Talkin. *A Robust Algorithm for Pitch Tracking (RAPT).* In W. B. Kleijn and K. K. Paliwal, editors, Speech Coding and Synthesis, chapter 14, pages 495–518. Elsevier Science B. V., 1995.

[Temperley01] D. Temperley. The cognition of basic musical structures. MIT Press, 2001.

[Terhardt74] E. Terhardt. *Pitch, consonance, and harmony.* J. Acoust. Soc. Am., vol. 55, no. 5, pages 1061–1069, May 1974.

[Tolonen00] T. Tolonen and M. Karjalainen. *A Computationally Efficient Multipitch Analysis Model.* IEEE Transactions on Speech And Audio Processing, vol. 8, no. 6, November 2000.

[Viitaniemi03] T. Viitaniemi, A. Klapuri and A. Eronen. *A Probabilistic Model For The Transcription Of Single-Voice Melodies.* In H. Huttunen, A. Gotchev and A. Vasilache, editors, Proceedings of The 2003 Finnish Signal Processing Symposium, Finsig'03, numéro 20 in TICSP Series, pages 59–63, May 2003.

[Viterbi67] A. Viterbi. *Error bounds for convolutional codes and an asymptotically optimum decoding algorithm.* IEEE Transactions on Information Theory, vol. 13, no. 2, pages 260–269, April 1967.

[Witten91] I. H. Witten and T. C. Bell. *The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression.* IEEE Transactions on Information Theory, vol. 37, no. 4, pages 1085–1094, July 1991.

[Young89] S. J. Young, N. H. Russell and J. H. S. Thornton. *Token Passing: a Simple Conceptual Model for Connected Speech Recognition Systems.* Technical report, Cambridge University Engineering Department, July 1989.

A Appendix: Detailed results for different feature sets

Simulation results when using key estimation and note bigrams are represented in tables A.1, A.2, and A.3 for feature sets $\{x_\Delta, \nu\}$, $\{x_\Delta, \nu, m\}$, and $\{x_\Delta, \nu, a, m\}$, respectively. The features are the pitch difference x_Δ, voicing ν, phenomenal accent a, and the metrical accent m. The symbols used in the tables are the following.

K, number of note-model states

η, number of GMM components in observation likelihood distributions

E_f, frame error percentage

E_n, note error percentage

K \ η	1	2	3	4	5	6	criterion
1	90.4	16.3	16.4	16.3	16.4	15.9	E_f
	64	19.9	18.8	18.6	18.6	18.6	E_n
2	74.6	12.1	12.4	12.5	12.1	12	E_f
	62.9	15.3	15	15.2	15.2	15.1	E_n
3	73.6	13.8	14.1	13.8	13.8	13.8	E_f
	61.9	14.8	15.5	15	14.9	14.9	E_n
4	19.8	10.8	10.7	10.6	10.6	10.5	E_f
	18.9	10.6	10.9	10.8	10.7	10.7	E_n
5	18.6	11	10.7	10.4	**10.3**	10.4	E_f
	17.3	10.8	10.8	10.5	**10.3**	10.4	E_n

Table A.1: Error rates for the feature set $\{x_\Delta, \nu\}$.

K \ η	1	2	3	4	5	6	criterion
1	90.4	16.3	16.4	16.1	15.7	15.7	E_f
	64	19.9	18.8	18.4	18.4	18.4	E_n
2	74.5	11.9	17.6	14.2	13.9	13	E_f
	63	15	14	13.3	14.8	15.5	E_n
3	18.9	10.5	11.6	10.4	**10.3**	10.4	E_f
	17.5	9.9	11.3	10.3	**10.2**	9.9	E_n
4	18.7	**12.2**	13.3	11.9	11.9	12.4	E_f
	17.9	**9.4**	11	10.1	10.1	10.1	E_n
5	11.4	13.3	13.2	12.1	12.5	12.9	E_f
	10.9	11.3	10.9	10.1	10.7	10.9	E_n

Table A.2: Error rates for the feature set $\{x_\Delta, \nu, m\}$.

K \ η	1	2	3	4	5	6	criterion
1	90.4	15.2	16.2	15.5	16	15.7	E_f
	64	19.4	18.4	18.1	18.5	18.2	E_n
2	73.8	11.9	14.2	13.7	13.1	13.1	E_f
	61.8	15	16.9	16.8	15.9	16.1	E_n
3	17.9	**9.1**	9.2	9.7	10.1	10.1	E_f
	18	**10.1**	10.1	10.5	10.8	10.7	E_n
4	18.2	9.4	**9.7**	9.8	10.4	10.2	E_f
	18.5	10.4	**9.9**	9.9	10.4	9.9	E_n
5	11.5	10.9	10.8	10.2	10.1	10.2	E_f
	11.9	11.7	11.6	11.1	10.7	10.9	E_n

Table A.3: Error rates for the feature set $\{x_\Delta, \nu, a, m\}$.