Automatic Transcription of Audio Signals

Master of Science Thesis

May 2004

Student

Jiří Vass
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Measurement
tel: +420 737 111030
e-mail: `vassj@fel.cvut.cz`
web: `http://measure.feld.cvut.cz`

Supervisors

Ing. Radim Špetík
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Circuit Theory
tel: +420 2 2435 2049
e-mail: `radim.spetik@email.cz`
web: `http://amber.feld.cvut.cz`

Hadas Ofir
Technion - Israel Institute of Technology
Department of Electrical Engineering
Signal and Image Processing Laboratory
e-mail: `hadaso@siglab.technion.ac.il`
web: `http://www-sipl.technion.ac.il`

Abstract

This thesis is concerned with automatic transcription of monophonic audio signals into the MIDI representation. The transcription system incorporates two separate algorithms in order to extract the necessary musical information from the audio signal. The detection of the fundamental frequency is based on a pattern recognition method applied on the constant Q spectral transform. The onset detection is achieved by a sequential algorithm based on computing a statistical distance measure between two autoregressive models. The results of both algorithms are combined by heuristic rules eliminating the transcription errors. Finally, new criteria for evaluation are proposed and applied on transcription results of several musical recordings.

Keywords: music transcription, pitch detection, fundamental frequency tracking, onset detection, monophonic audio

Abstrakt

Tato diplomová práce se zabývá automatickou transkripcí jednohlasých hudebních signálů do formátu MIDI. Transkripční systém zahrnuje dva samostatné algoritmy nezbytné pro získání hudební informace z audio signálu. Detekce základní harmonické složky je založena na metodě hledání vzorů ve spektrální transformaci s konstantním činitelem jakosti Q. Detekce začátků not je dosaženo pomocí sekvenčního algoritmu založeného na výpočtu statistické metriky mezi dvěma autoregresními modely. Výsledky obou algoritmů jsou sloučeny pomocí heuristických pravidel odstraňujících chyby transkripce. Závěrem jsou navržena nová kritéria pro vyhodnocování, která jsou použita na výsledky transkripce několika hudebních nahrávek.

Klíčová slova: transkripce hudby, detekce základní harmonické, detekce nestacionarit, jednohlasá hudba

Acknowledgements

First of all, I would like to thank to my parents and relatives for their immense support during my studies.

I would like to thank to Ing. Radim Špetík for supervision of this thesis. Next, I thank very much to Prof. Ing. Pavel Sovka, CSc. and Doc. Ing. Petr Pollák, CSc. for excellent education, as well as for help in the design of new criteria.

The biggest thanks goes to my initial supervisor Hadas Ofir who is the co-author of the proposed system. A special thanks goes also to Nimrod Peleg for choosing the very best topic of my summer project in the SIPL laboratory. I am also very grateful to Heikki Jokinen, Pekka Kumpulainen and Anssi Klapuri for stimulating my interest in signal processing and audio applications in particular.

Thanks goes also to my friends Michal Olexa and Václav Vozár for valuable hints concerning DSP, Matlab and TeX. And as a nice Czech proverb says: "the best at the end", I would like to thank to my ♡ Zuzka Lenochová.

This work is dedicated to all musicians I have had the opportunity and the pleasure to play with…

Poděkování

V první řadě bych chtěl poděkovat svým rodičům a příbuzným za nesmírnou podporu v průběhu studia.

Rád bych poděkoval Ing. Radimu Špetíkovi za odborné vedení této diplomové práce. Dále velmi děkuji Prof. Ing. Pavlu Sovkovi, CSc. a Doc. Ing. Petru Pollákovi, CSc. za vynikající výuku a také pomoc při návrhu nových kritérií.

Největší dík patří mé počáteční odborné vedoucí Hadas Ofir, jež je zároveň spoluautorkou navrženého systému. Zvlaštní dík patří též Nimrodu Pelegovi za výběr toho nejlepšího možného tématu pro můj letní projekt v laboratoři SIPL. Jsem také velmi vděčný Heikki Jokinenovi, Pekka Kumpulainenovi a Anssi Klapurimu za vznícení mého zájmu o zpracování signálů se zaměřením na audio aplikace.

Děkuji též svým přátelům Michalu Olexovi a Václavu Vozárovi za cenné rady tykající se DSP, Matlabu a TeXu. A podle úsloví "to nejlepší nakonec" bych rád poděkoval své ♡ Zuzce Lenochové.

Tato práce je věnována všem muzikantům, se kterými jsem měl tu možnost a čest si zahrát…

Čestné prohlášení

Prohlašuji na svou čest, že jsem zde uvedenou diplomovou práci vypracoval samostatně, pouze za odborného vedení Ing. Radima Špetíka a při psaní diplomové práce jsem nepoužil jiných informačních zdrojů než zde uvedených.

Jiří Vass

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Characterization of the Problem

Automatic transcription of music is a task of converting a particular piece of music into symbolic representation by means of a computational system. Symbolic representation is generally depicted using the *standard music notation* which consists of *notes* characterized by specific frequency and duration. From the transcription point of view, music can be classified as *polyphonic* and *monophonic*. The former consists of multiple simultaneously sounding notes, whereas the latter contains only a single note at each time instant, such as a saxophone solo or singing of a single vocalist.

Automatic transcription of music is related to several fields of science, including *Musicology*, *Psychoacoustics*, and *Computational Auditory Scene Analysis* (CASA). It belongs to the *music content analysis* discipline which consists of other audio research topics, such as rhythm analysis, instrument recognition, and sound separation. It has been studied since 1970s.

1.2 Literature Review

The state-of-the-art in music transcription is focused on the polyphonic transcription, since the monophonic transcription is considered as practically solved [Klapuri1998], [Martins2001]. However, it represents an important case which should be treated separately with much stricter demands on the transcription quality, which still seems to be relatively limited for polyphonic transcribers. Extensive review of published polyphonic systems can be found in [Klapuri1998].

Since monophonic music share various properties with speech, many algorithms suitable for the music transcription purposes originate in speech processing [Rabiner1976], [Hess1983], [Andre-Obrecht1986],[Medan1991]. Recent works in monophonic music transcription explore the potential of the wavelet transform [Cemgil1995a], [Cemgil1995b], [Jehan1997], time-domain techniques based on autocorrelation [Bello2002], and probabilistic modelling using Hidden Markov Models [Ryynänen2004]. In addition to that, [Bořil2003] developed a simple and robust algorithm for real-time MIDI conversion, referred to as *DFE algorihtm* (Direct Time Domain Fundamental Frequency Estimation). This system performs separate monophonic analysis of a signal from each guitar string, and therefore illustrates that monophonic transcribers can be used in special polyphonic transcription systems.

1.3 Applications

Applications of automatic transcription systems are numerous, though limited due to insufficient reliability and robustness. The following list presents the potential areas of interest.

- *Computer music applications*

  Music transcription system is a useful tool for composers and musicians, since it provides means to easily analyze and edit the music recordings. It is especially attractive for the real-time transcription of sounds to musical score.

- *Coding of audio signals*

  Conversion of signal samples to symbolic representation significantly reduces the amount of data, and can be therefore used for the compression purposes. An example method is the *structured audio* (SA) coding described in the MPEG-4 Standard.

- *Mobile technology*

  Reliable transcription systems could be commercially applied in cellular phones to automatically create monophonic or polyphonic *Ringtones*. Such feature would allow customers to record their own musical compositions by a cellular phone and transmit the MIDI files via the Internet or the GSM network.

- *Machine perception*

  Analogically to computer vision, the ability of computers to hear music would improve the interaction between humans and systems with artificial intelligence.

- *Music teaching*

  Future transcription systems could be used in training of singers and solo instrument players, as well as assist in ear training of novice musicians. Such systems would compare the exact musical notation with the performance of an artist and objectively evaluate the performance quality.

1.4 Organization of the thesis

This thesis inclines to be oriented more practically than theoretically, and thus briefly explains only the essential background information and refers the reader to other publications, often available online. For this reason, it omits a separate theoretical chapter and defines the necessary terms "on-the-fly" during the description of the transcription system.

This thesis is organized as follows. Chapter 2 gives an overview of the MIDI standard. Chapter 3 presents the implemented solution. In Chapter 4, new criteria for evaluation are proposed and the transcription results are presented. Finally, Chapter 5 summarizes the accomplishments.

Chapter 2

The MIDI Standard

2.1 MIDI Introduction

The *Musical Instrument Digital Interface* (MIDI) provides a standardized means of conveying musical performance information as electronic data. It has been accepted and utilized by musicians and composers since its conception in 1983, and is nowadays widely used for communication between sound cards, musical keyboards, sequencers, and other electronic instruments. A complete description of the MIDI protocol is defined in the *MIDI 1.0 Specification* established and updated by the MIDI Manufacturers Association [MMA2004].

The main advantage of MIDI is data storage efficiency: a typical MIDI sequence requires approximately 10 Kbytes of data per minute of sound. Contrary to WAV files, which contain digitally sampled audio in the PCM format, the MIDI files consist of *MIDI messages* which can be understood as special instructions for synthesizers to generate the real sounds. These messages thus provide very efficient *symbolic representation* of music. Moreover, the MIDI files are also editable, allowing the music to be rearranged or even composed interactively.

2.2 MIDI Basics

The MIDI architecture consists of three main components: hardware interface (connector), a communication protocol (language), and a distribution format (Standard MIDI File).

2.2.1 MIDI Hardware Interface

The MIDI interface of each instrument is generally provided by three MIDI connectors, labeled IN, OUT, and THRU. The only approved MIDI connector is a 5-pin DIN connector. The physical *MIDI channel* is divided into 16 logical channels, each capable of transmitting MIDI messages from and to a single musical instrument.

2.2.2 MIDI Communication Protocol

The MIDI data stream is a unidirectional asynchronous bit stream at 31,25 Kbits/s with 10 bits transmitted per byte (a start bit, 8 data bits, and one stop bit). The MIDI protocol is composed of *MIDI messages* in a binary form; each message is formed by an 8-bit status byte, followed by one or two data bytes.

MIDI messages are processed in *real time*, i.e. when a MIDI synthesizer receives a *note-on message*, it plays the appropriate sound, and stops this sound when the corresponding *note-off message* is received. Similarly, when a key is pressed on the musical instrument keyboard, the note-on message is immediately generated, as well as the note-off message is generated when this key is then released. Therefore, no timing information is transmitted with the MIDI messages in the real time applications.

2.2.3 Standard MIDI Files

However, in order to store the MIDI data as a data file, *time-stamping* of the MIDI messages must be performed to guarantee playback in a proper time sequence. In other words, each message is assigned a value of time in the SMPTE format (hours : minutes : seconds : frames) and the resulting specification is referred to as the *Standard MIDI File* (SMF) format. In addition to that, the SMF specification further defines three MIDI file formats, because the MIDI sequencers can generally manage multiple MIDI data streams, called *tracks*.

- MIDI Format 0 stores all MIDI data in a single track, although it may represent several musical parts at different MIDI channels.
- MIDI Format 1 stores MIDI data as a collection of tracks (up to 256), each musical part separated in its own track.
- MIDI Format 2, which is relatively rare and often not supported, can store several independent songs.

Since this work is concerned with the monophonic audio, only the MIDI Format 0 is used and the terms *track* and *MIDI channel* are interchangeable. It should also be noted that the MIDI files can be converted by the *MIDI File Format Conversion Utility* provided by [Glatt2004].

Finally, a MIDI file can also be understood as a "musical version" of an ASCII text file, except that it contains binary data. Indeed, [Glatt2004] also offers the *MIDI File Dis-Assembler Utility* converting a MIDI file to a readable text, which can then be edited in a text editor and converted back to a modified MIDI file.

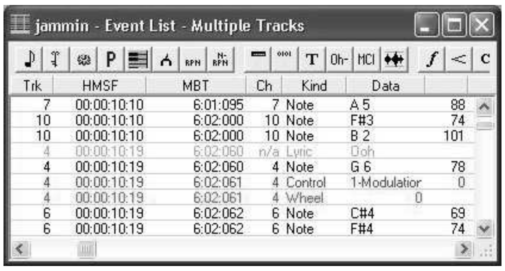
2.3 MIDI File Representations

Although there are many different types of MIDI messages, this work is concerned only with the *note-on* and *note-off messages* carrying the musical notes data, and hence comprising most of the traffic in a typical MIDI data stream. The remaining MIDI messages are applied mainly for hardware tasks, such as selecting which instrument to play, mixing and panning sounds, and controlling various aspects of electronic musical instruments.

At the highest level, MIDI messages are classified into *system messages* and *channel messages*. The latter are also called *MIDI events* consisting of such events as a Note, Pitch Wheel, and Aftertouch. On the other hand, a MIDI file can also store other information related to a musical performance using special *non-MIDI events*, including Tempo, Lyric, Track and Device Name, Time and Key Signature, Copyright, and others (see [Martins2001, Appendix B] for a detailed description).

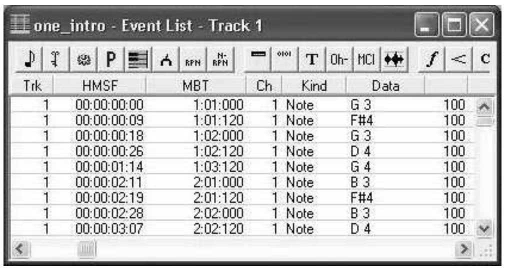
Both MIDI and non-MIDI Events form an *event list*, which can be viewed by commercial audio software programs, such as [Cakewalk2004]. Figure 2.1(a) shows the event list for the case of polyphonic music (multiple tracks, MIDI Format 1), whereas Figure 2.1(b) shows the monophonic case (a single track, MIDI Format 0). The former gives some examples of MIDI and non-MIDI events in the *Kind* column, and also illustrates the varying values in the *Ch* and *Trk* columns (MIDI channel and track, respectively), typical for multi-instrumental MIDI files. Notice that polyphony and chords can easily be achieved by placing several *note events* at a specific time instant (for example, C#4 and F#4 at 00:00:10:19).

As can be seen in the comparison of Figure 2.1(b) with Figure 2.2, the event list provides the MIDI data representation equivalent to the *note staff* and the *piano roll*. Note staff expresses the MIDI data from the musical point of view, while the piano roll is commonly used as a simple and practical depiction of the transcription results (see Chapter 4). Figure 2.1(b) also shows


jammin - Event List - Multiple Tracks

Trk	HMSF	MBT	Ch	Kind	Data	
7	00:00:10:10	6:01:095	7	Note	A 5	88
10	00:00:10:10	6:02:000	10	Note	F#3	74
10	00:00:10:10	6:02:000	10	Note	B 2	101
4	00:00:10:19	6:02:060	n/a	Lyric	Ooh	
4	00:00:10:19	6:02:060	4	Note	G 6	78
4	00:00:10:19	6:02:061	4	Control	1-Modulatior	0
4	00:00:10:19	6:02:061	4	Wheel	0	
6	00:00:10:19	6:02:062	6	Note	C#4	69
6	00:00:10:19	6:02:062	6	Note	F#4	74

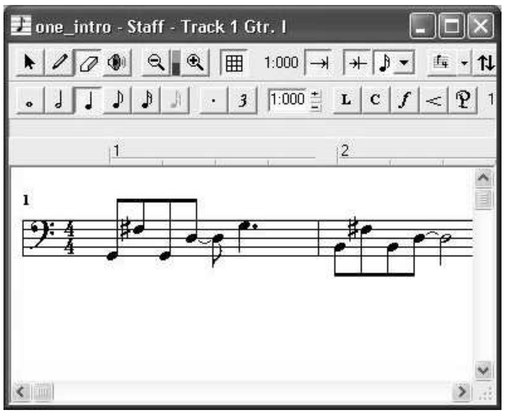
(a) polyphonic


one_intro - Event List - Track 1

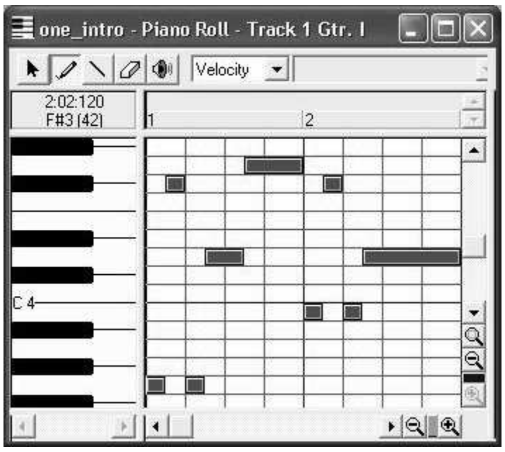
Trk	HMSF	MBT	Ch	Kind	Data	
1	00:00:00:00	1:01:000	1	Note	G 3	100
1	00:00:00:09	1:01:120	1	Note	F#4	100
1	00:00:00:18	1:02:000	1	Note	G 3	100
1	00:00:00:26	1:02:120	1	Note	D 4	100
1	00:00:01:14	1:03:120	1	Note	G 4	100
1	00:00:02:11	2:01:000	1	Note	B 3	100
1	00:00:02:19	2:01:120	1	Note	F#4	100
1	00:00:02:28	2:02:000	1	Note	B 3	100
1	00:00:03:07	2:02:120	1	Note	D 4	100

(b) monophonic

Figure 2.1: Event list of a typical MIDI file



(a) Note Staff



(b) Piano Roll

Figure 2.2: Alternative representations of a monophonic MIDI file

Onset time [s]	Note duration [s]	MIDI note number [-]	MIDI key velocity [-]	MIDI channel [-]
0.0878	0.1372	55	127	0
0.2250	0.2035	57	110	0
0.4285	0.1518	59	90	0
...	...	...	...	...
2.0867	0.3731	79	100	0

Table 2.1: Simplified event list

that only the note events are necessary to describe the tones of a digitally sampled audio, and therefore, event list can be reduced to a simplified version shown in Table 2.1.

2.4 Event List Parameters

As can be observed, each line in Table 2.1 contains an individual note event comprising the following event list parameters:

2.4.1 Onset Time

This parameter characterizes the exact time of the beginning of a note, referred to as *onset*, which is directly connected to the note-on MIDI message. Analogically, the *offset* is the termination of a note, and is connected to the note-off MIDI message. For a monophonic signal, it can naturally be assumed that an onset of a new note causes the offset of the previous note.

2.4.2 Note Duration

The duration of a note is defined as a time difference between the onset and the offset of a particular note. It is important to mention that *rests* (musical pauses) are therefore not explicitly defined, since they are automatically produced when the onset time plus the duration of the current note is less than the onset time of the next note.

2.4.3 MIDI Note Number

Each *MIDI note number* (integer between 0 and 127) defines a fixed value of frequency corresponding to a certain note (see the *Data* column in Figure 2.1(b)). For instance, the note A2 (A is the note name, number 2 indicates the second *octave*) has the frequency of 440 Hz and is defined by the MIDI note number 69 (see Figure 3.2).

As further explained in Section 3.1, the individual MIDI note numbers are calculated from the detected values of *pitch* in an audio signal. Pitch is commonly defined as perceived *fundamental frequency*, although these terms are usually interchanged [Klapuri1998].

2.4.4 MIDI Key Velocity (Volume)

The *MIDI key velocity* parameter (0 to 127) is derived from the velocity with which a piano hammer hits a string, and therefore expresses the note volume (loudness) or sound intensity. Nevertheless, some MIDI devices are velocity insensitive.

2.4.5 MIDI Channel

The *MIDI channel* parameter (0-15) gives the index of the logical channel used for transmission of the MIDI message corresponding to the current event. This parameter is not of our interest, since the monophonic audio requires only a single channel, and is hence by default set to zero.

Chapter 3

Music Transcription System

This chapter provides a detailed description of the proposed music transcription system. As implies from the preceding Chapter, the simplified event list incorporates all data necessary to create a monophonic MIDI file. Therefore, given an input audio signal, the essential objective of the transcription system is to extract the event list parameters discussed in Section 2.4. This task is divided into the individual building blocks as follows (see Figure 3.1).

The *Pre-processing* block normalizes the signal to the unit power, and inserts silence before the beginning and after the end of the signal. The *Pitch Detection* block is mainly responsible for tracking the fundamental frequency in the signal, but also contributes to the determination of the note onset and offset times, which is the principal task of the *Detection of Events* block (whose output can conversely affect the fundamental frequency tracking). Since none of these blocks yield ideal results, the *Estimation of Power* block is added to provide supportive data for both the event detection,

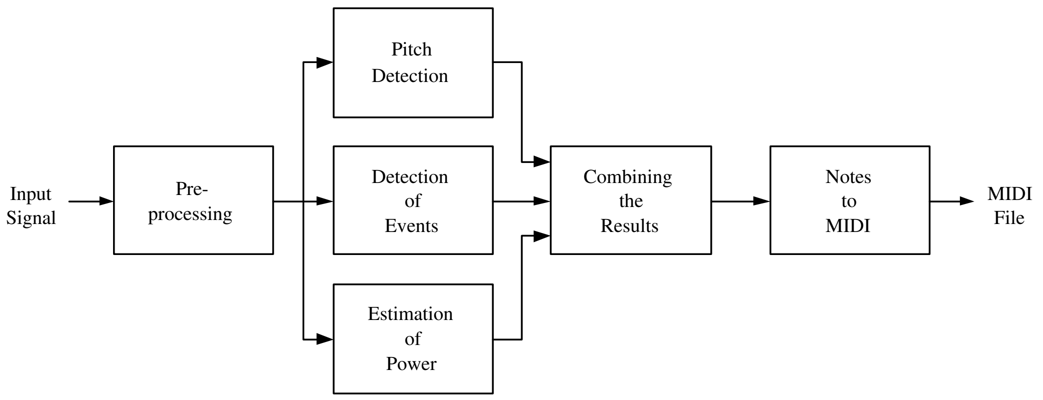


Figure 3.1: Transcription system overview.

and the calculation of the MIDI volume. Finally, the *Combining the Results* block processes the outputs of the three preceding sections, and generates the complete event list in an appropriate format.

Fortunately, a Matlab code performing the final *Notes to Midi* conversion is available on the Internet [Cemgil2004], so the MIDI standard must have been studied only to the limited extent presented earlier. For implementation details, refer to the Appendix A.5.

3.1 Pitch Detection

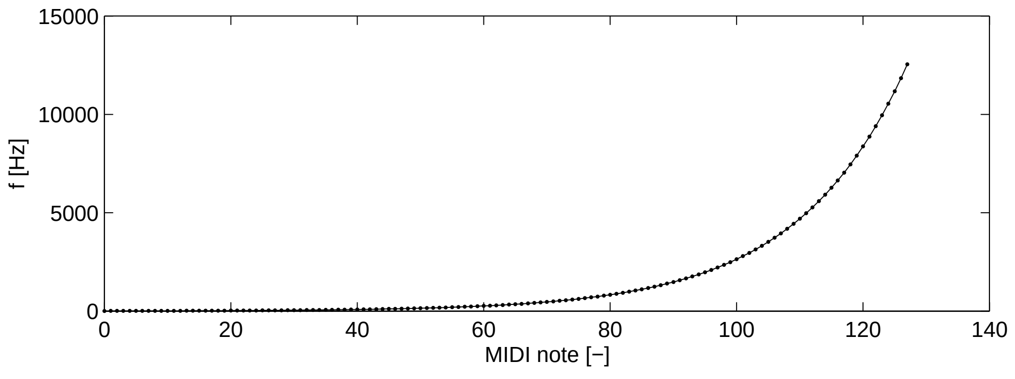
Great effort was devoted to explore various algorithms of the pitch detection, in order to find the best solution suitable for this thesis. It was soon revealed that the monophonic transcription is practically a solved issue [Klapuri1998], hence the remaining problem was to discover some advanced method feasible in a relatively short time.

Numerous pitch detection techniques were originally developed for the speech applications [Rabiner1976], but unfortunately, many of them are less appropriate for the audio processing purposes due to the reasons summarized in the conclusion of [Cemgil1995a]. Finally, a solution based on the *Constant Q Transform* (CQT) was selected, since it offers the best results in terms of the time-frequency trade-off among the methods compared in [Cemgil1995a]. In addition to that, the primary articles explaining this transform and the Matlab implementation are available [Brown2004].

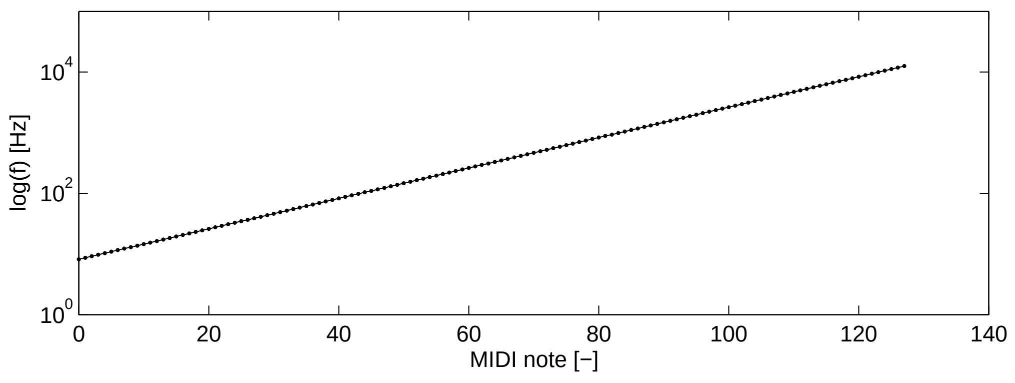
3.1.1 MIDI Frequencies

As can be observed in Figure 3.2(a), the MIDI standard defines only 128 possible musical notes corresponding to an exponentially increasing set of frequencies in the linear scale (geometric series). This concept was borrowed from the classical western music theory, in which all frequencies chosen to form the musical scales are also spaced exponentially. Such scales are referred to as *equally tempered scales* [Klapuri1998] or *equal temperament scales* [Martins2001].

The principal reason for this choice is that the human ear is characterized by the logarithmic frequency resolution, i.e. the absolute resolution is finer at low frequencies, and coarser at high frequencies. Therefore, exponentially spaced musical frequencies are perceived with constant quality within the whole frequency range, as can be seen in Figure 3.2(b).



(a)



(b)

Figure 3.2: Spacing of musical frequencies

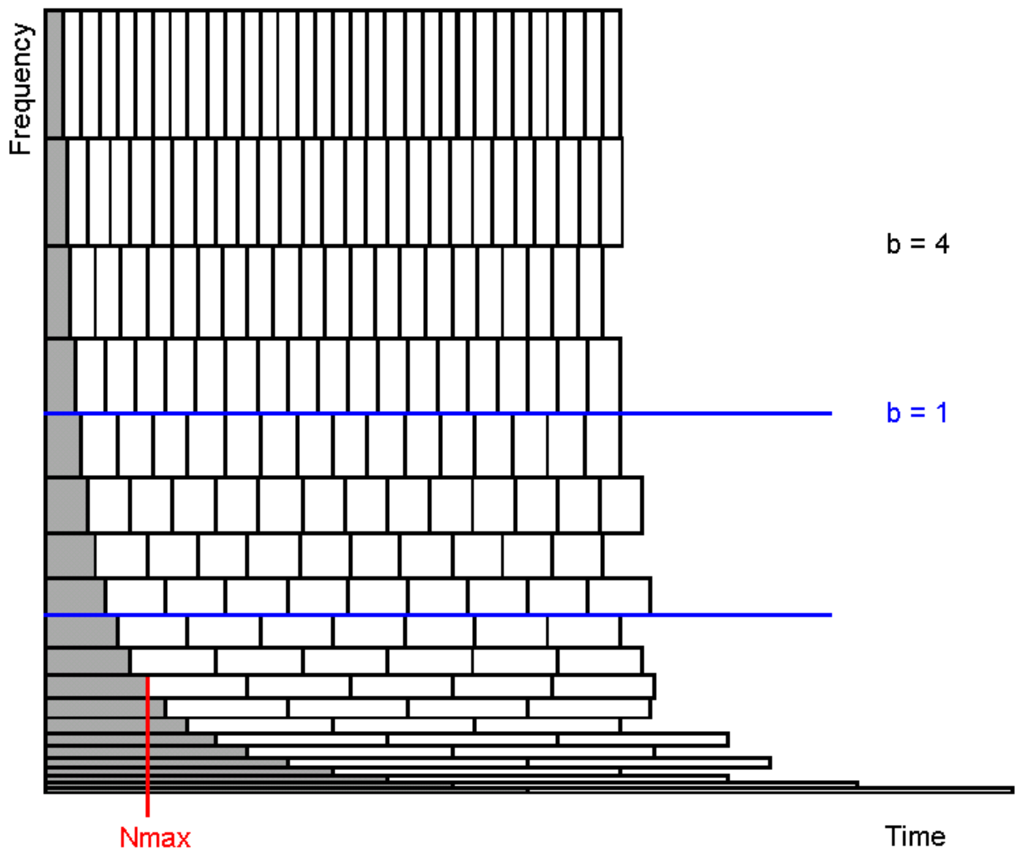


Figure 3.3: Tiling of the time-frequency plane by the CQT

3.1.2 Time-Frequency Representation

Since the musical frequencies form a geometric series, it is desirable to represent the signal with spectral transform corresponding to a filter bank with center frequencies spaced in the same manner. Such transform was introduced in [Youngberg1979], [Brown1991] and is referred to as *Constant Q Transform* (CQT). Similarly as in the DFT, the frequency range is divided into frequency bins, each represented by a bandpass filter with a center frequency f_k and filter bandwidth Δf_k. However, the *CQT bins* are geometrically spaced, resulting in variable resolution at different octaves.

Figure 3.3 shows the tiling of the time-frequency plane by the CQT filter bank and reviews the crucial properties of the transform.

1. Center frequencies f_k increase exponentially (i.e. form a geometric series).

Quantity	CQT	DFT
Center frequencies	exponential in k $f_k = f_0 \left(\sqrt[b]{2}\right)^k$	linear in k $f_k = k \cdot \Delta f$
Window length	variable = $N(k)$	constant = N
Filter bandwidth	variable = f_k/Q	constant = f_s/N
Resolution	constant = Q	variable = k

Table 3.1: Comparison of CQT and DFT.

2. Filter bandwidths Δf_k become narrower towards low frequencies, and wider towards high frequencies.

3. With b being the *number of filters per octave*, the CQT filter bank is equivalent to $1/b^{th}$ octave filter bank, which shows its close relationship with the wavelet transform [Cemgil1995a]. Hence, when $b = 1$ (see the blue lines), the CQT is identical to the standard octave filter bank, and the *quality factor* $Q = f_k/\Delta f_k$ equals to unity.

   The western music divides each *octave* into 12 intervals, called *semitones*, corresponding to $b = 12$. For this reason, $b = 24$ is typically used to obtain "twice as good" (i.e. *quarter-tone*) resolution, necessary to reliably detect semitones anywhere in the musical frequency range.

4. Variable window length $N(k)$
   Since the windows at low frequencies become extremely long, a fixed parameter of maximum window length N_{max} (see the red line) is used to reduce these windows in order to preserve sufficient time resolution, since N_{max} is also used as a window length for the segmentation of the signal, similarly as in STFT. This improvement of CQT is called the *Modified CQT* (MCQT) and was introduced in [Brown1993].

Previous observations are summarized in Table 3.1. Note that $f_k = f_0 \left(\sqrt[b]{2}\right)^k$ is indeed a formula for geometric series with quotient $\sqrt[b]{2}$, and that the spacing of all frequencies depends on the minimal center frequency f_0. For further theoretical background, refer to [Brown1991] and [Blankertz2004].

Figures 3.4 and 3.5 provide a visual comparison between the CQT and the DFT transforms for the time-frequency analysis of the sound from a violin playing G major scale from G3 (196 Hz) to G5 (784 Hz). Both CQT representations show very clearly the spectral content of the signal, so the note changes can easily be observed (due to the obvious separation of the fundamental frequencies), as well as the *formant* region around 3000 Hz. Also note

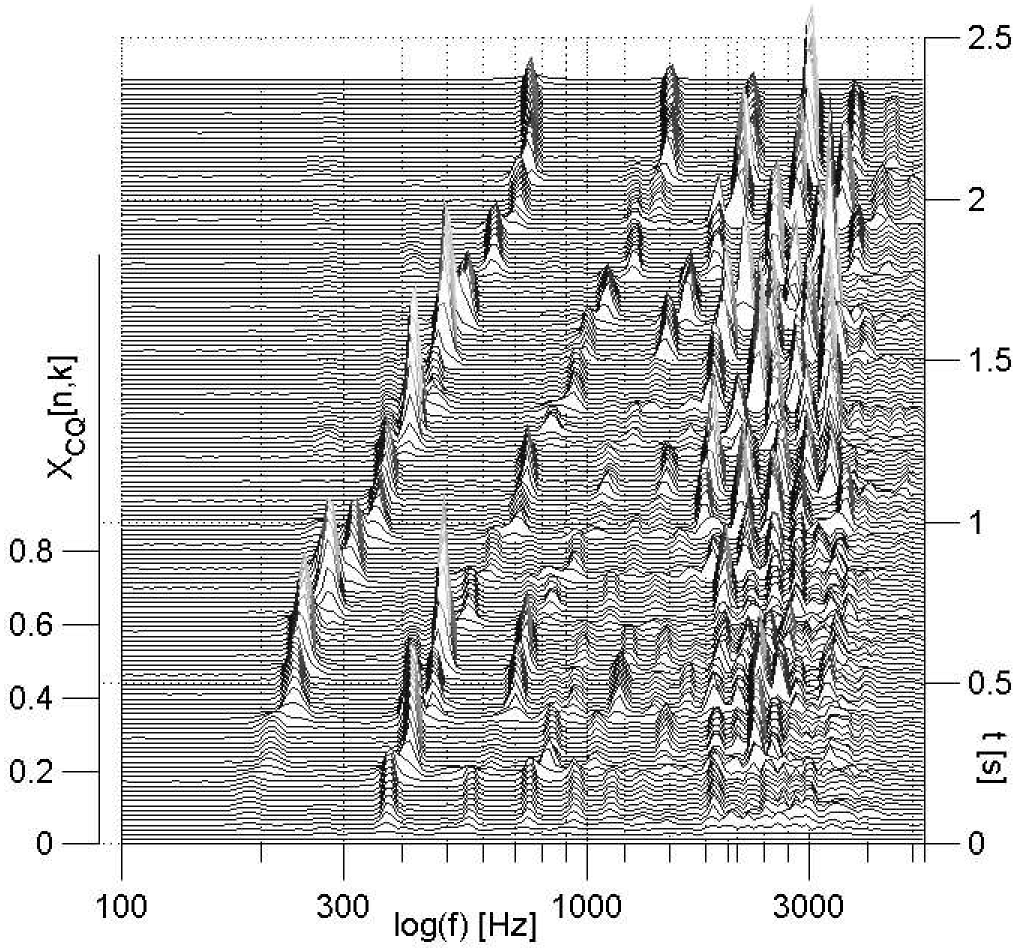
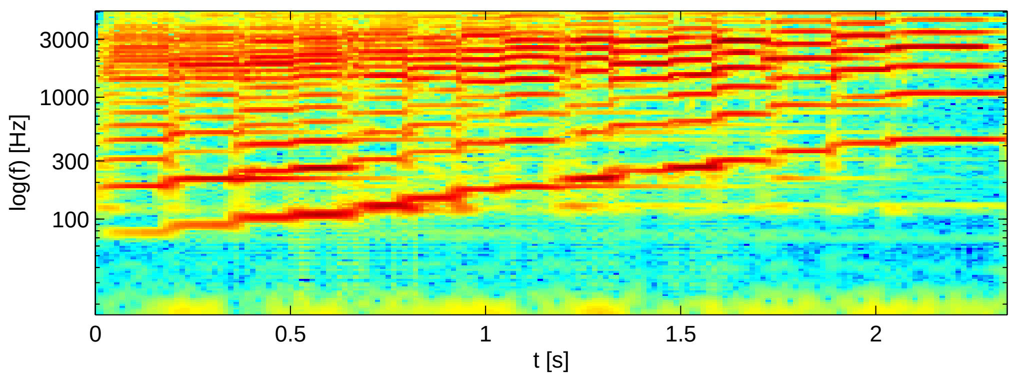
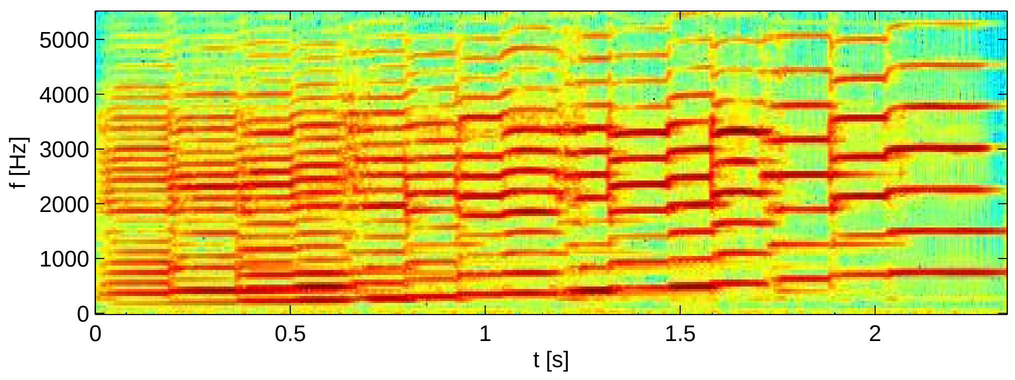


Figure 3.4: CQT plot of a violin sound



(a)



(b)

Figure 3.5: Comparison between spectra of a violin sound: (a) CQT spectrogram, (b) STFT spectrogram

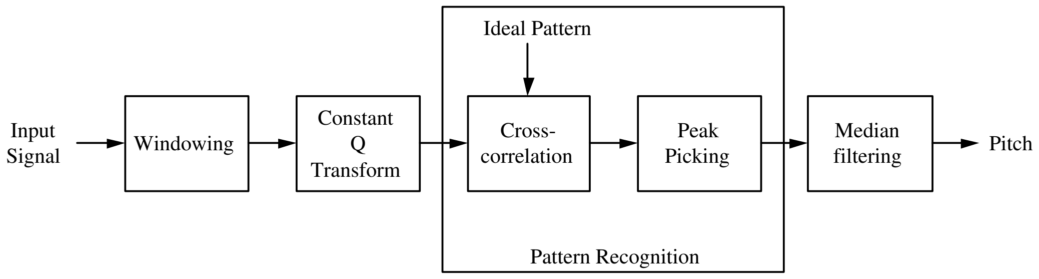


Figure 3.6: Pitch detection system overview

that the first two notes (G3 and A3) have an almost undetectable fundamental frequency. The STFT spectrogram, on the other hand, demonstrates that the DFT indeed does not map to musical frequencies efficiently, providing very little information at low frequencies, and too much information at high frequencies.

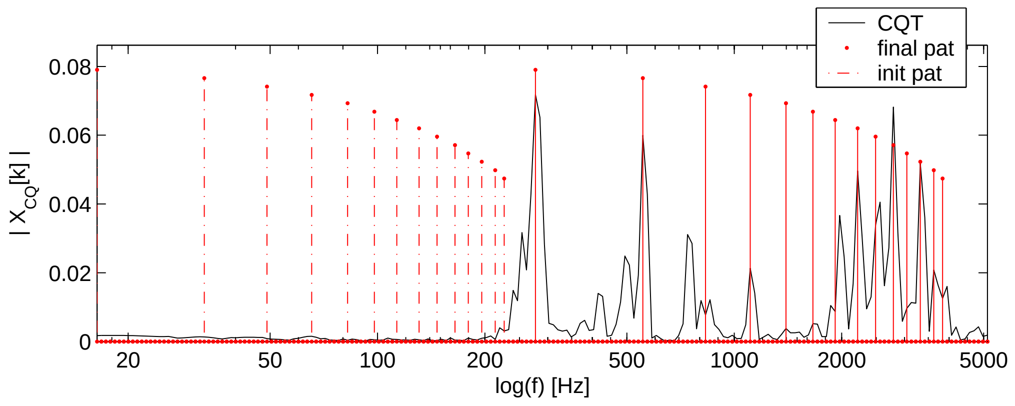
3.1.3 Pitch Detection System

After briefly explaining the motivation for using the Constant Q Transform, a CQT-based pitch detector can now be presented. Figure 3.6 depicts the basic building blocks representing the internal structure of the *Pitch Detection* block in Figure 3.1. The main principle is adopted from [Brown1992a], which was subsequently improved in [Brown1993] using phase changes of the Fourier transform. However, the technique described in the latter article did not prove advantageous, so it was removed from the overall structure.

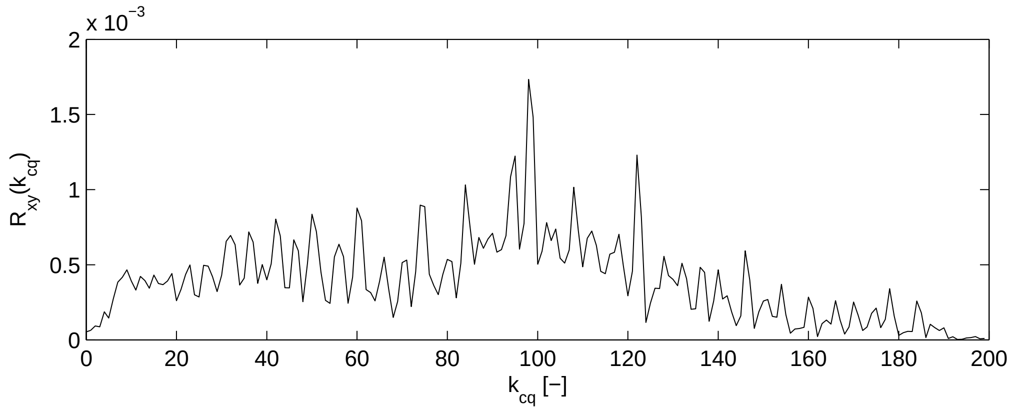
The pitch detection system operates as follows: first of all, the input signal is divided into overlapping segments (using the Hamming window by default), which are then transformed by the CQT block. The CQT spectrum is then processed by a frequency tracker based on *pattern recognition* (PR), computing an estimate of the fundamental frequency within the given segment (see Section 3.1.4). Finally, all detected values of pitch are nonlinearly smoothed by median filters in order to eliminate gross errors (see Section 3.1.5).

3.1.4 Frequency Tracking

The CQT spectral components form a *constant pattern* when plotted against the logarithmic frequency axis (see [Brown1991, Figure 1]). Indeed, it can easily be shown that the spacing between harmonics is independent of the fundamental frequency:



(a)



(b)

Figure 3.7: Frequency tracking based on pattern recognition

$$\log(f_m) - \log(f_n) = \log\left(\frac{f_m}{f_n}\right) = \log\left(\frac{m \cdot F_0}{n \cdot F_0}\right) = \log\left(\frac{m}{n}\right) \tag{3.1}$$

In other words, the relative positions between harmonics are constant for all musical notes, and only the absolute positions depend on the fundamental frequency. This property can be employed to determine the fundamental frequency as a maximum value of the cross-correlation function between an ideal (theoretical) pattern and the pattern in the actual CQT spectrum, as outlined in Figure 3.6.

Figure 3.7 presents the results of this procedure for a particular signal segment. Figure 3.7(a) shows the CQT amplitude spectrum of the segment, as well as the initial and the final positions of the ideal pattern (chosen to consist of 14 harmonics). The initial position is created with the fundamental situated at zero frequency. As the cross-correlation function is calculated for increasing values of *lags* (each corresponding to a CQT component or bin), it can be visualized as generating a new pattern with higher position of the fundamental, or equivalently, shifting the original pattern by one CQT bin "to the right". Please note that this explanation is somewhat popular, and the correct mathematical interpretation is to be found in [Brown1992a]. After performing such operation within the whole frequency range, the fundamental is determined as the frequency corresponding to the CQT at the maximum of the cross-correlation function. The values at higher harmonics in the ideal pattern are linearly decaying in order to emphasize the fundamental frequency, and thus reduce the *octave errors* during the pattern recognition [Brown1993].

3.1.5 Post-processing of Results

As depicted in Figure 3.6, the results are finally post-processed by the median filtering block performing a nonlinear smoothing operation. This block consists of two successive median filters (of orders 3 and 5, respectively) referred to as *combination median smoother* ([Rabiner1975], [Hess1983]). In this case, it is inherently capable of removing 3 isolated gross errors in a 5-point interval.

Figure 3.8(a) illustrates the frequency tracking results for the signal of the violin (introduced in Figure 3.4 and 3.5). Each point here corresponds to an individual value of pitch determined in each segment. As can be seen, the median 3 filter first removes single discontinuities (red crosses), followed by the median 5 filter eliminating the remaining 2-point deviations (black crosses), resulting in an almost ideal "stairs" (blue dots).

Figure 3.8b depicts the transcription of a piano sound obtained by resynthesis of a MIDI file back to an audio WAV file. Contrary to the previous case, this transcription involves additional *error* values, corresponding to silent regions in the input signal. Each of these errors is caused by failure in pattern recognition, indicated by a maximum of cross-correlation function at a *negative* value of the CQT component k_{cq} (see Figure 3.7(b)). To enable further calculations, the negative values are replaced by ones, resulting in erroneous frequencies located in the "valleys" in Figure 3.8(b). Such frequencies are therefore omitted by the median filters, and are subsequently utilized in the final decision procedure (see Section 3.4.2).

Notice that the piano sound in Figure 3.8(b) plays several notes at two distinct frequencies 57 Hz and 68 Hz. While the tones of the former frequency can readily be recognized due to the "error valleys", the latter frequency is

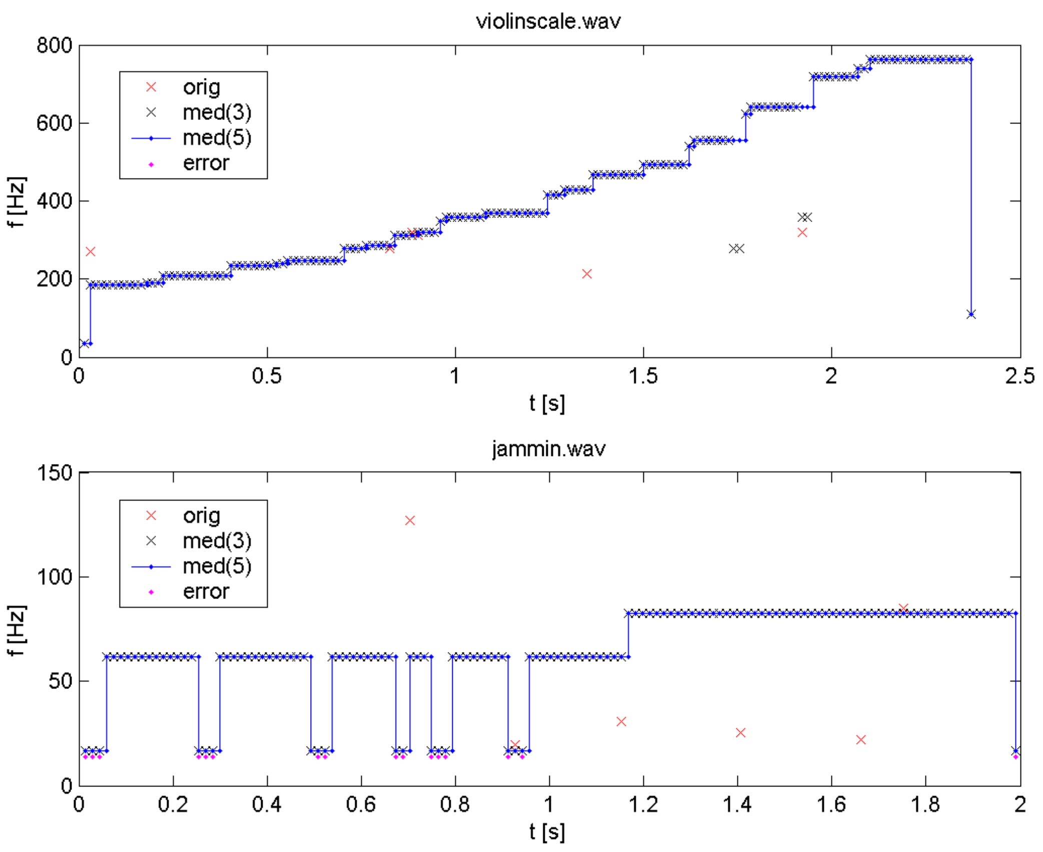


Figure 3.8: Application of median filtering: (a) real violin sound, (b) synthesized piano sound

merged into a single line. However, it can be shown that the nearby "red crosses" do not represent gross errors of the frequency tracking, but in fact, they clearly indicate each new note. From this point of view, it might seem that the median smoothing excludes some information about the note detection provided by the CQT. On the other hand, such method for the note detection would be very inaccurate and problematic (as shown in Figure 3.8(a)), and thus a separate onset detection algorithm must be implemented.

3.2 Detection of Events

The algorithm for detection of acoustic changes (*events*) is based on the article [Basseville1983], further improved in [Andre-Obrecht1988], and discussed in [Jehan1997]. Jehan received the algorithm written by Andre-Obrecht in Fortran, and included the C-language implementation in his MSc. Thesis. This code was then rewritten and much work has been made to remove the old-fashioned programming style, and to optimize it as a Matlab MEX-file. Great advantage of this algorithm is the *statistical time-domain approach* performed on a sample-by-sample basis, thus providing very accurate locations of the onset and offset times.

3.2.1 AR Modeling

The main idea is to model the signal by an *autoregressive* (AR) model, and to use *test statistic* to sequentially detect changes in the parameters of this model. The signal is assumed to be described by a string of homogeneous units, each characterized by the AR model $M = (\underline{a}, \sigma_e^2)$ of the following form (p is the *order* of the model).

$$\hat{x}[n] = -\sum_{i=1}^{p} a_i x[n-i] + e[n] \tag{3.2}$$

1. Vector of AR coefficients:

$$\underline{a} = (a_1, a_2, \dots a_p) \tag{3.3}$$

   Since the AR coefficients represent the LPC spectrum of a signal, this part of the model is responsible for tracking abrupt *changes in spectral characteristics*, typical for the transient parts (*attacks*) of the signal. Equivalently, transients are characterized in the time domain by a very

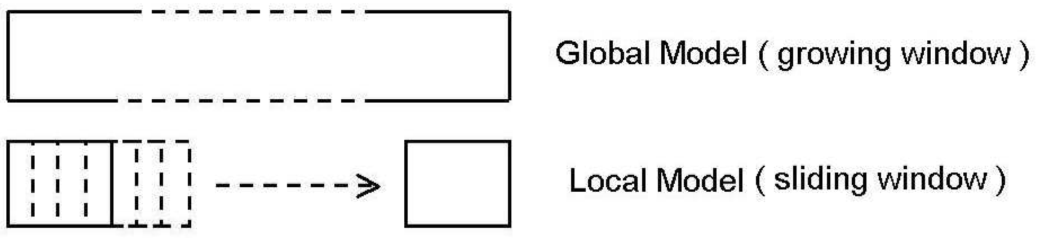

Global Model (growing window)
Local Model (sliding window)

Figure 3.9: Two AR models

steep increase in amplitude (see Figure 3.11), also requiring rapid changes in the AR model (when interpreted as a *linear predictor* of the consecutive sample).

2. Variance of the prediction error signal:

$$\operatorname{var}(e[n]) = \sigma_e^2 = \psi_e^2 = P_e = \alpha \tag{3.4}$$

This part of the model describes *changes in power* of the signal, as implied by the above formula. The error signal $e[n]$ is assumed to be a zero-mean white noise, with the variance σ_e^2 constant inside the homogeneous units.

In general, there are three possible cases for detection of an event: a change in AR coefficients, a change in power, and a change in both. Therefore, it has become necessary to perform a separate power analysis (see Section 3.3) in order to distinguish between the cases, and classify the events into onsets and offsets.

3.2.2 Divergence Test

Although classical methods based on one AR model exist, [Basseville1983] demonstrated the advantages of the statistical test based on two AR models, between which a suitable *distance measure* is monitored. This distance measure is referred to as *conditional Kullback's divergence*, hence the name of the algorithm (which will be often used in the sequel).

Figure 3.9 shows the locations of the two AR models M_0 and M_1. The former is a *global* (or *long-term*) *model*, whereas the latter is a *local* (or *short-term*) *model*. As can be seen, the global model is represented by a *growing window*, beginning at the first sample of the signal, while the local model is symbolized by a *sliding window* of the constant length L. For this reason, the global model is recursively updated using a sample-by-sample growing memory *Burg algorithm* (described in [Basseville1982, Appendix I]), while the local model is calculated using the *autocorrelation method.* The identification of both models is followed by the computation of their distance measure, and a new *event* is announced whenever the distance exceeds a specific threshold value. That is, the two AR models disagree due to some abrupt change during the last L samples of the signal. Finally, the exact time instant of this change is estimated by the *Hinkley's* (stopping-time) *test*, providing a sophisticated detection with very small delay. The overall process is summarized in the flow chart in Figure 3.10.

It is worth mentioning that the short-term window is not sliding throughout the entire signal waveform, due to the second "decision" in the flowchart, requiring at least L newly analyzed samples (after detection of a previous event). Therefore, the sliding window can reappear only after satisfying this condition, resulting in a certain minimal time between each two successive events.

3.2.3 Parameters Selection

As elaborately discussed in [Basseville1983] and [Andre-Obrecht1988], the segmentation results are strongly dependent on two primary parameters:

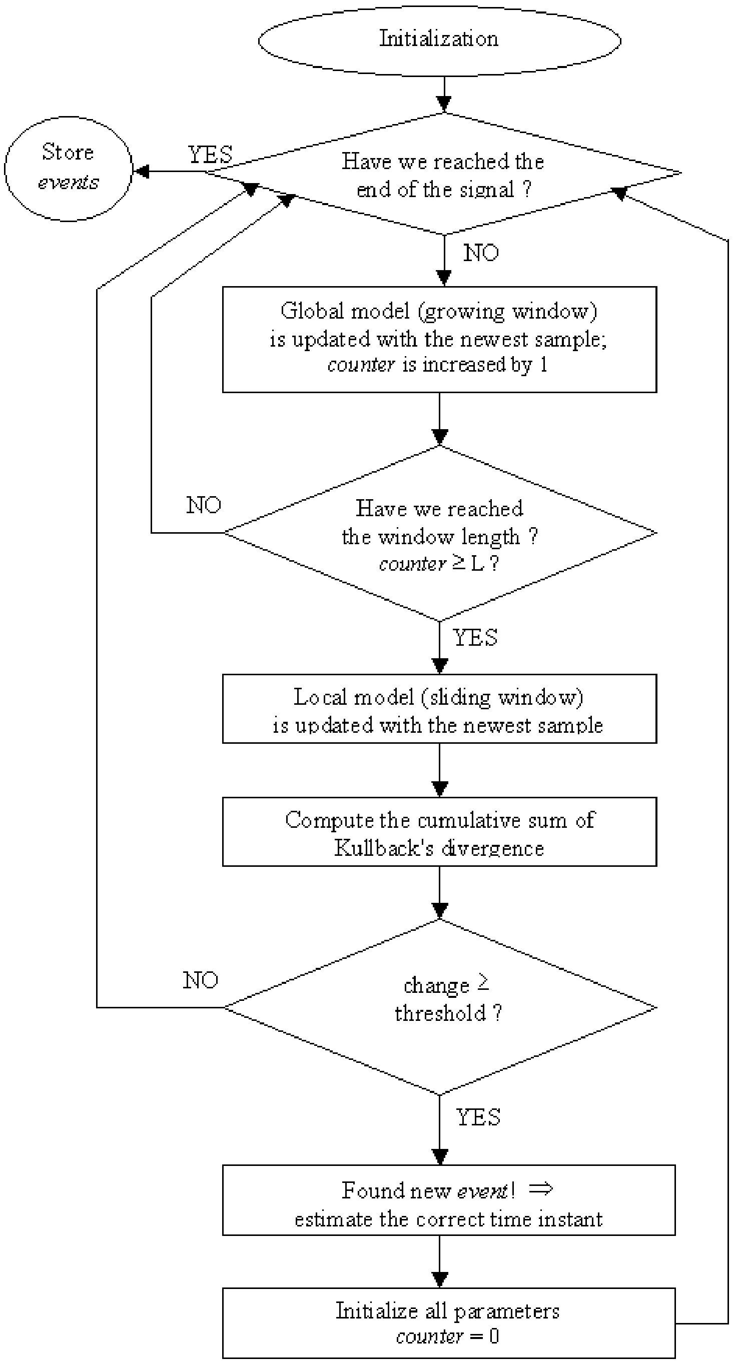
a) Sliding window length L

As stated above, the short-term model is identified using the autocorrelation method based on solving the *Yule-Walker equations* $\mathbf{R}\cdot\underline{a} = -\mathbf{R_p}$, where $\underline{a}$ is the vector of autoregressive coefficients and $\mathbf{R}$ is the autocorrelation matrix. This matrix consists of autocorrelation coefficients, which can be calculated using the *biased estimate* of the autocorrelation function:

$$\hat{R}[k] = \frac{1}{L} \sum_{n=0}^{L-k-1} x[n] \cdot x[n+k] \quad \text{for } k = 0, 1, \ldots p \qquad (3.5)$$

b) AR model order p

In theory, the model order should not be a fixed number, since the optimal value may progress in time. Fortunately, this is was not a severe problem in the tested signals, since a relatively high order was used to assure detection of all possible events. Generally, the higher is the model



Initialization
Have we reached the end of the signal ?
YES
Store *events*
NO
Global model (growing window) is updated with the newest sample; *counter* is increased by 1
Have we reached the window length ? *counter* ≥ L ?
NO
YES
Local model (sliding window) is updated with the newest sample
Compute the cumulative sum of Kullback's divergence
change ≥ threshold ?
NO
YES
Found new *event*! ⇒ estimate the correct time instant
Initialize all parameters *counter* = 0

Figure 3.10: Event detection process.

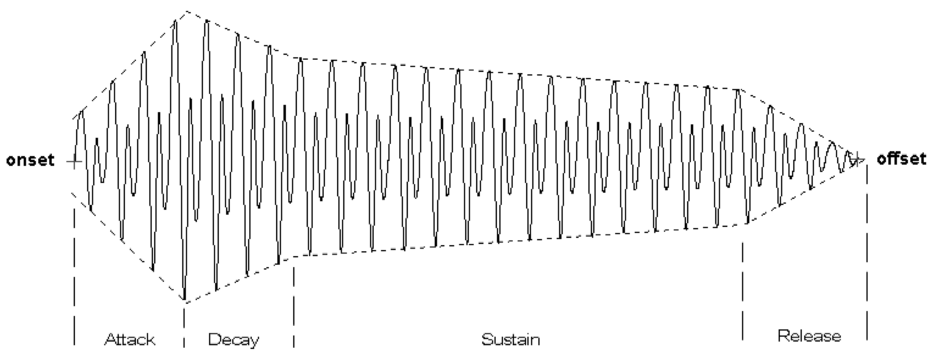


Figure 3.11: Attack-Decay-Sustain-Release (ADSR) model of a typical note.

order, the more events are detected, and thus the higher probability of undesired *oversegmentation*. Nevertheless, such phenomenon can be allowed to a certain extent, because the output of the event detector is corrected by the *Combining the Results* block (described in Section 3.4). Accordingly, the order 30 is selected as a default, contrary to [Jehan1997] who experimented with very low values (2, 4, and 6) for real-time calculations.

3.3 Estimation of Power

3.3.1 ADSR Model

As discussed in Sections 3.1 and 3.2, the main motivation for estimating the signal power is the fact that the results of the divergence test provide no information about the origin of the detected event. The test locates not only the majority of onsets and offsets, but for some notes also an *attack*, *decay*, and *release*, defined in the *ADSR envelope model* shown in Figure 3.11.

The attack is defined as a region between the zero amplitude and the peak, and exhibits the steepest ascent of power in the ADSR model. It is then followed by a *decay* region, also attended by an abrupt change in power, which may cause a detection of an incorrect event. Such error can be avoided by accepting this event only when its signal power is reasonably low to regard it as an offset (i.e. the end of a *release* region). Therefore, there is a need for determination of local power minima, which are then used to eliminate the main errors of the divergence test, such as:

- *False alarms* due to inadequate length of the sliding window
- Mistakes arising from oversegmentation

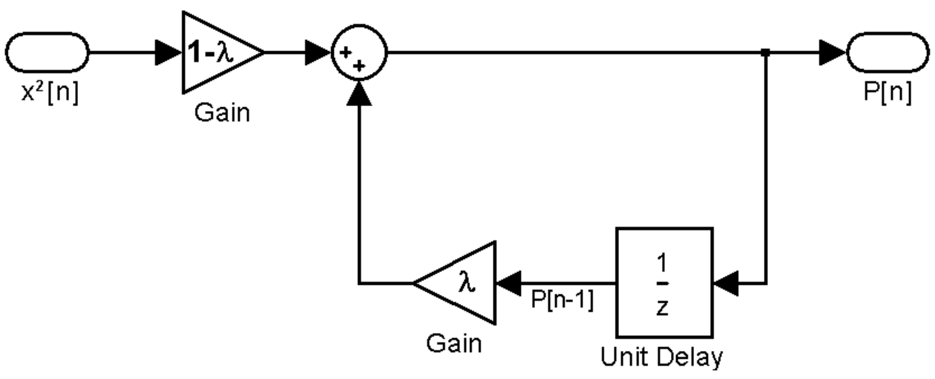


Figure 3.12: Computation of the recursive estimate of power

- Abrupt changes in power not connected with the note onsets or offsets

3.3.2 Leaky Integrator

Because the musical signals are non-stationary by its nature, it is necessary to track the signal power in time. As discussed in [Sovka2003], there are two main approaches for power estimation, namely the *block estimate*, and the *recursive estimate.* After several experiments, it emerged that more suitable results are obtained by the latter system, referred to as the *leaky integrator.* As can be seen in Figure 3.12, it is a simple first-order recursive digital filter (RDF), characterized by the following difference equation:

$$P[n] = \lambda \cdot P[n-1] + (1+\lambda) \cdot x^2[n] \tag{3.6}$$

where λ is the *forgetting factor* and $x^2[n] = p[n]$ is the *instantaneous power* of the input signal, which is being smoothed by the filter. The difference equation can be Z transformed and rewritten to obtain the system transfer function and the impulse response:

$$H(z) = \frac{P(z)}{H(z)} = \frac{1-\lambda}{1-\lambda z^{-1}} \tag{3.7}$$

$$h[n] = (1-\lambda) \cdot \lambda^{-n} \quad \text{for } n = 0, 1, \ldots \infty \tag{3.8}$$

Figure 3.13 displays the exponential impulse response of the IIR filter $H(z)$, as well as its *time constant* τ defined in this case as:

$$\tau = -\frac{1}{\ln \lambda} \qquad \Leftrightarrow \qquad \lambda = \exp(-1/\tau) \tag{3.9}$$

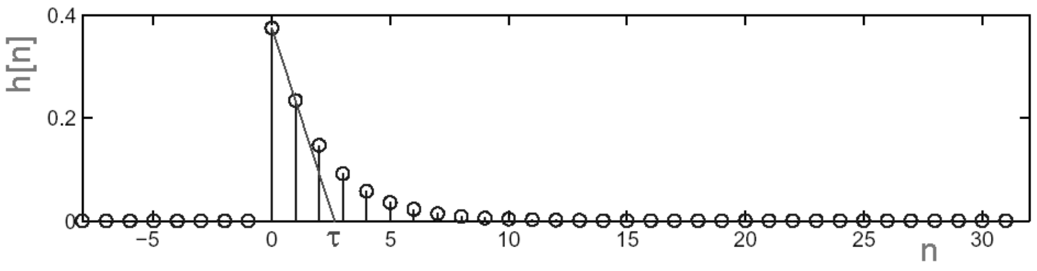


Figure 3.13: Impulse response of the leaky integrator

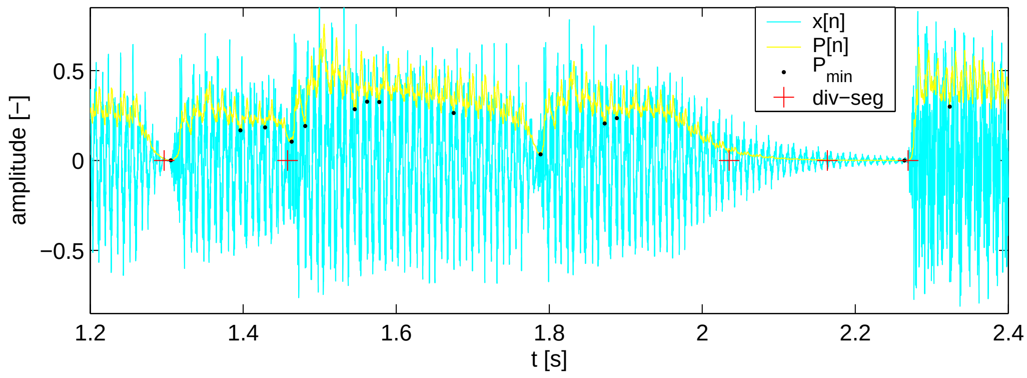


Figure 3.14: Estimate of power with the desired local minima.

It was decided to use the value of τ equivalent to the window length N_{max} for MCQT (see Figure 3.3), which corresponds to $\tau = \frac{N_{\text{max}}}{f_s} \cdot 1000 \approx 93$ ms and $\lambda = 0.99$. Such selection provides the desired tracking of amplitude changes in a signal at the expense of rather fluctuating estimate of power, which tends to follow many unimportant variations. Nevertheless, this drawback is insignificant, since we are primarily interested in accurate time localization of local power minima P_{min} depicted in Figure 3.14.

Let us first repeat that each sample of the incremental estimate of power $P[n]$ is the output of the IIR(1) leaky integrator with the exponential impulse response $h[n]$. Since this response is highly asymmetrical, the power $P[n]$ steeply increases during attacks in the signal $x[n]$, and slowly decreases afterwards, which creates an evident local minimum of power at each note onset. Moreover, it can clearly be observed that these power minima always coincide with the correct events detected by the divergence test (marked in Figure 3.14 by red crosses), and can thus be used for reliable elimination of the errors discussed in Section 3.3.1.

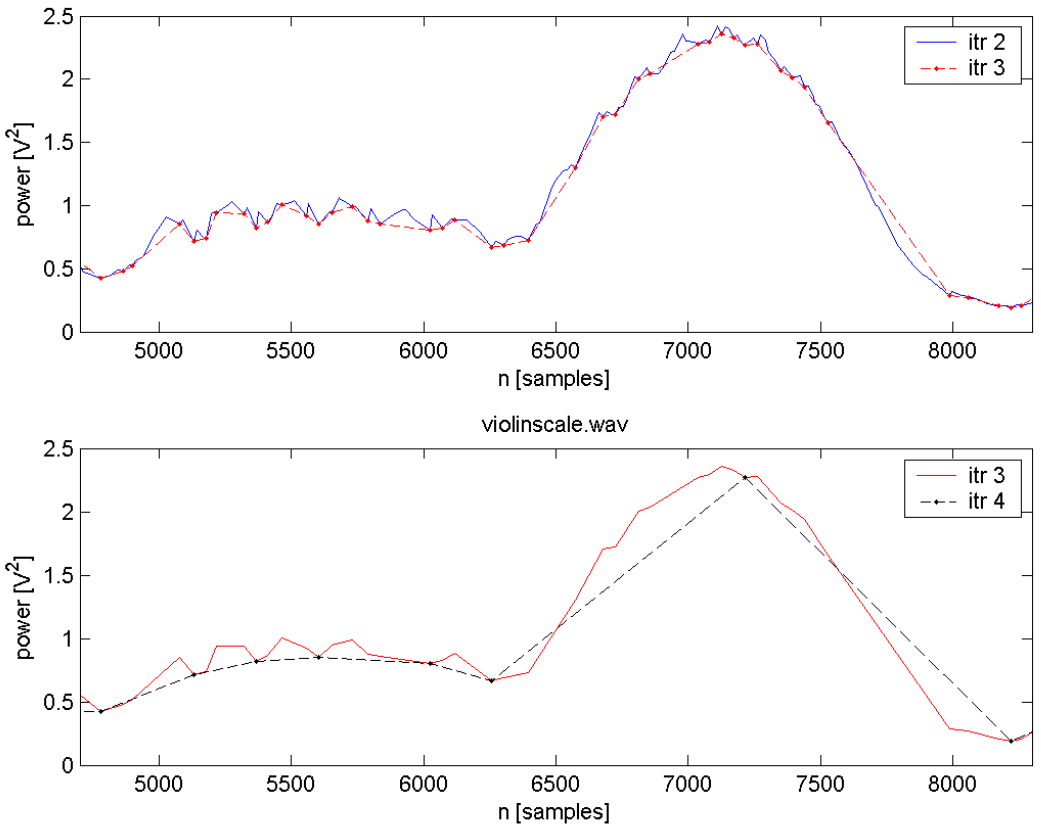


Figure 3.15: Principle of the iterative smoothing function.

3.3.3 Iterative Smoothing Function

This section describes a function smoothing the fluctuations in the power $P[n]$ in order to locate its most significant minima P_{min}. The function is based on the mathematical principle defining a minimum as a point preceded by a negative value of derivation, and succeeded by positive derivation. Since this principle inherently finds all local minima, the procedure must be performed iteratively, as illustrated in Figure 3.15.

The upper picture displays the solid blue curve representing the smoothed version of $P[n]$ already after two iterations. Then, all local minima are detected (shown as dots), and interconnected into a dashed line. Finally, the resulting line is depicted on the lower picture again as a solid curve, and the process is repeated. As you can notice, the curve visibly becomes *piecewise linear* with each new iteration, and also the amount of local minima is rapidly reduced. The process is stopped when this amount is less than the number of events from the divergence test, requiring typically three to five iterations (depending on the signal length).

Additionally, an analogical procedure is executed for detection of the most significant local maxima of power, each corresponding to an amplitude peak at the end of a note attack (see Figure 3.11). These maxima are also utilized in the final decision procedure, described in the next section.

3.4 Combining the Results

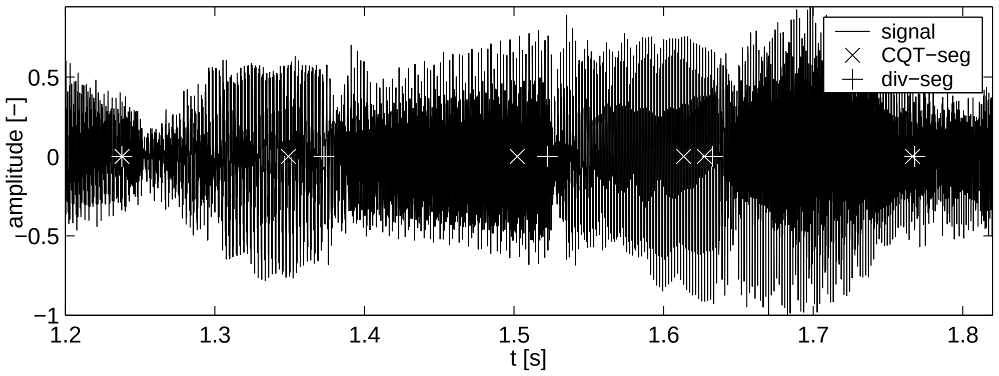
As introduced in Chapter 3, the purpose of the *Combining the Results* block is to produce a final event list based on the outputs of the three previous blocks. Principally, this block applies several heuristic rules to correctly choose the best candidate for the onset time and MIDI frequency of each note. These rules form an eliminative competition between two sets of candidates, and are the contents of Section 3.4.2.

3.4.1 CQT-based Segmentation

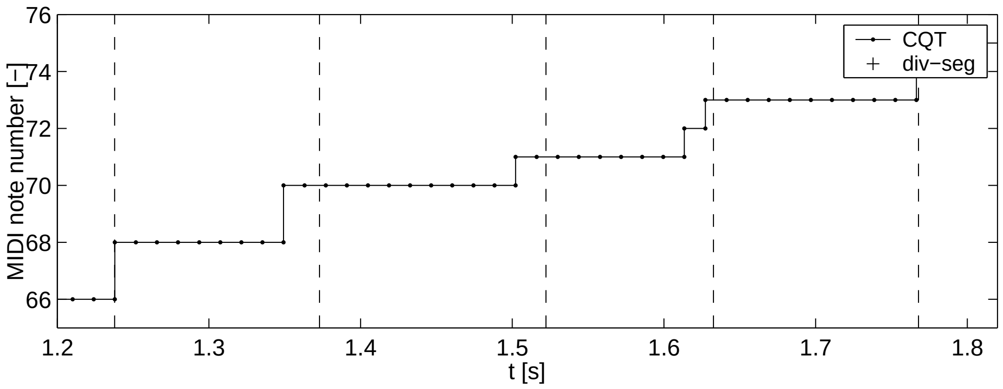
First of all, we will extend the results from Section 3.1.5 to create the first set of candidates. Figure 3.16(a) shows a portion of the violin sound presented earlier, whereas Figure 3.16(b) illustrates the frequency tracking results obtained from those in Figure 3.8(a) by quantizing the detected frequencies to MIDI note numbers.

Figure 3.16(b) graphically supports a straightforward assumption that every change in frequency causes a new note, so the CQT indeed provides elementary onset detection in the input signal $x[n]$. Such onsets constitute an initial set of note candidates, and are depicted in Figure 3.16(a) together with the candidates nominated by the divergence test. As can be observed, the segmentation based on the CQT seems to be rather inaccurate because of the time resolution dependent on the window length N_{max}. Furthermore, the overall pitch detection system tends to generate several false tones, such as the short note at approximately 1,62 s.

The divergence test, on the other hand, offers nearly exact localization of the signal changes owing to the sample-by-sample approach. However, it suffers many short-comings (see Section 3.3.1) causing the difficulty to distinguish between the *offset-onset pair* around 1,25 s, and two nearby onsets around 1,78 s (probably due to oversegmentation). Consequently, the errors of both algorithms must be eliminated in a competitive manner described in the sequel.



(a)



(b)

Figure 3.16: (a) CQT-based segmentation (b) MIDI frequency tracking

3.4.2 Eliminative Competition

This procedure makes a final decision on the correctness of candidates by the CQT and the divergence test. Let us for simplicity denote the former set of candidates as `cqtseg`, and the latter as `divseg`. Then, the competition can be divided into separate rounds or rules, each responsible of eliminating or accepting a specific subset of candidates.

Round One

The decisive rule applied in the first round can briefly be summarized as: "The winner is the nearest". Specifically, the algorithm sequentially processes the `cqtseg` candidates, and assigns to each member the nearest candidate (in samples) from the `divseg`. Any `cqtseg` member is automatically discarded when there are no `divseg` candidates in the region between the previous and the next `cqtseg` member. When two `cqtseg` members share the same `divseg` candidate as a winner, only the member corresponding to a longer note is preserved.

Indeed, such operations appear to remove most of the undesired candidates (shown in Figure 3.16), and effectively correct the onset times of the `cqtseg` candidates.

Round Two

As demonstrated in Figure 3.14, the correct `divseg` candidates are often located in the vicinity of power minima P_{min} preceding the note attacks. This property is thus the necessary condition of the second round, which allows an additional acceptance of the `divseg` candidates rejected in Round 1. It should be emphasized that such rule is capable of recognizing successive notes of the same frequency, and hence allows the time segmentation which would not be possible solely with the CQT approach.

Round Three

The third round is based on the observation that all monophonic audio signals contain an attack between each two consecutive onsets. In other words, the power $P[n]$ must have at least one local maximum P_{max} to consider such onsets as beginnings of two separate notes. This criterion represents the third eliminative rule illustrated in Figure 3.17.

As can be observed, this rule excluded the `divseg` candidate at approximately 1,14 s for the benefit of the candidate at 1,2 s. The former was chosen in Round 1 (as the closest to the `cqtseg` candidate), while the latter

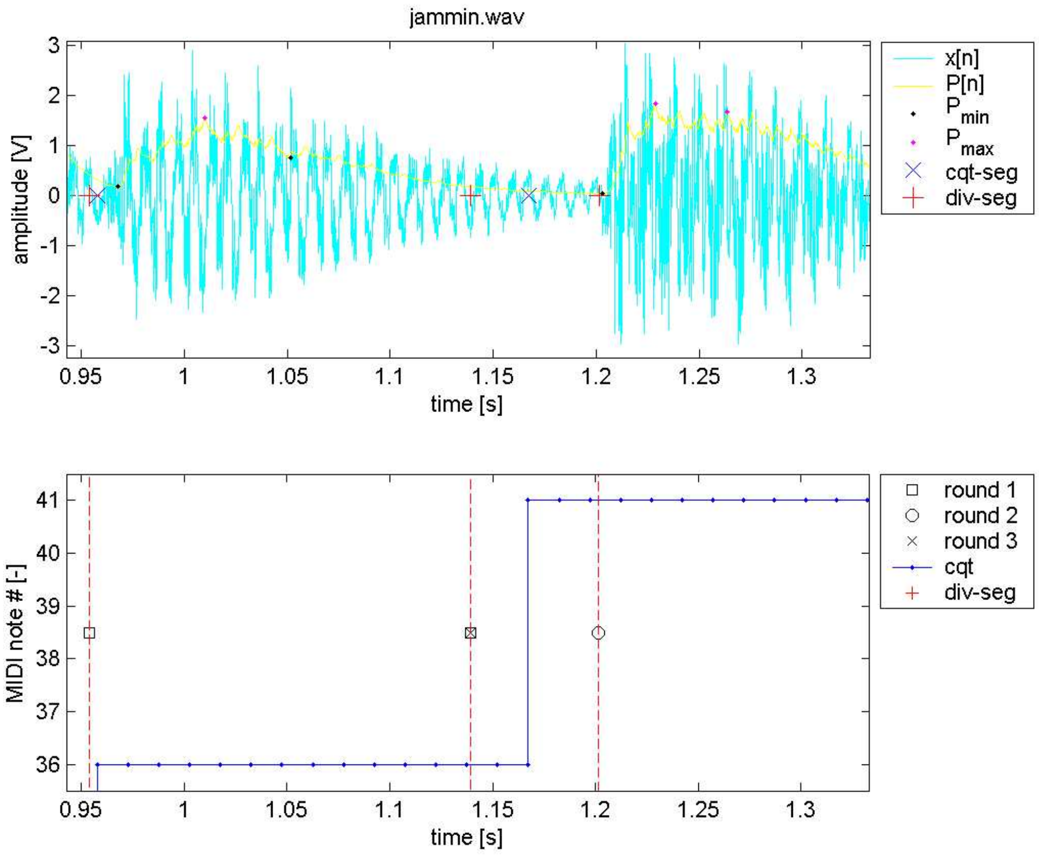


Figure 3.17: Principle of the eliminative competition

was accepted in Round 2 for satisfying the condition of proximity to a power minimum. Finally, the former candidate is removed because it violates the condition of Round 3, which was easily satisfied by the true note onsets at 1,2 s and 0,95 s (as can be seen in the upper figure).

Chapter 4

Simulation Results

This chapter presents the evaluation of the transcription system described in Chapter 3. Section 4.1 describes the criteria of transcription quality, Section 4.3 specifies the settings of transcription system parameters, Section 4.2 depicts the methods of creating the testing data, and finally Section 4.4 discusses the results based on several examples.

4.1 Transcription Quality Criteria

This section describes the criteria for evaluating the performance of the proposed transcription system. The criteria represent an attempt to numerically quantify the transcription results, since pure listening and word commentary is insufficiently informative. The new criteria were developed due to the absence of objective measures at the time of writing of this thesis. However, [Ryynänen2004] have recently published other useful criteria with some properties similar to those described in Section 4.1.3.

The criteria can be divided into two separate groups: *time-based* criteria and *note-based* criteria. The former is described in Section 4.1.1, whereas the latter is explained in Section 4.1.3.

4.1.1 Time-based Criteria

The time-based evaluation is inspired by the criteria in [Pollák2002], which were partially adopted from [Rosca2002]. These criteria originate in speech processing and were designed to evaluate the performance of *Voice Activity Detectors* (VAD). In our context, they serve to assess the accuracy of onset

OVF	OVerlap at the Front	OON	Overlap at ONset
OVB	OVerlap at the Back	OOF	Overlap at OFfset
TRF	TRuncation at the Front	TON	Truncation at ONset
TRB	TRuncation at the Back	TOF	Truncation at OFfset

Table 4.1: Terminology of the time-based criteria

and offset detection, which represents a parallel to the task of speech segmentation. Therefore, analogous criteria can simply be obtained by substituting the *Front* and the *Back* of a speech activity by the note *Onset* and *Offset*, respectively, as shown in Table 4.1.

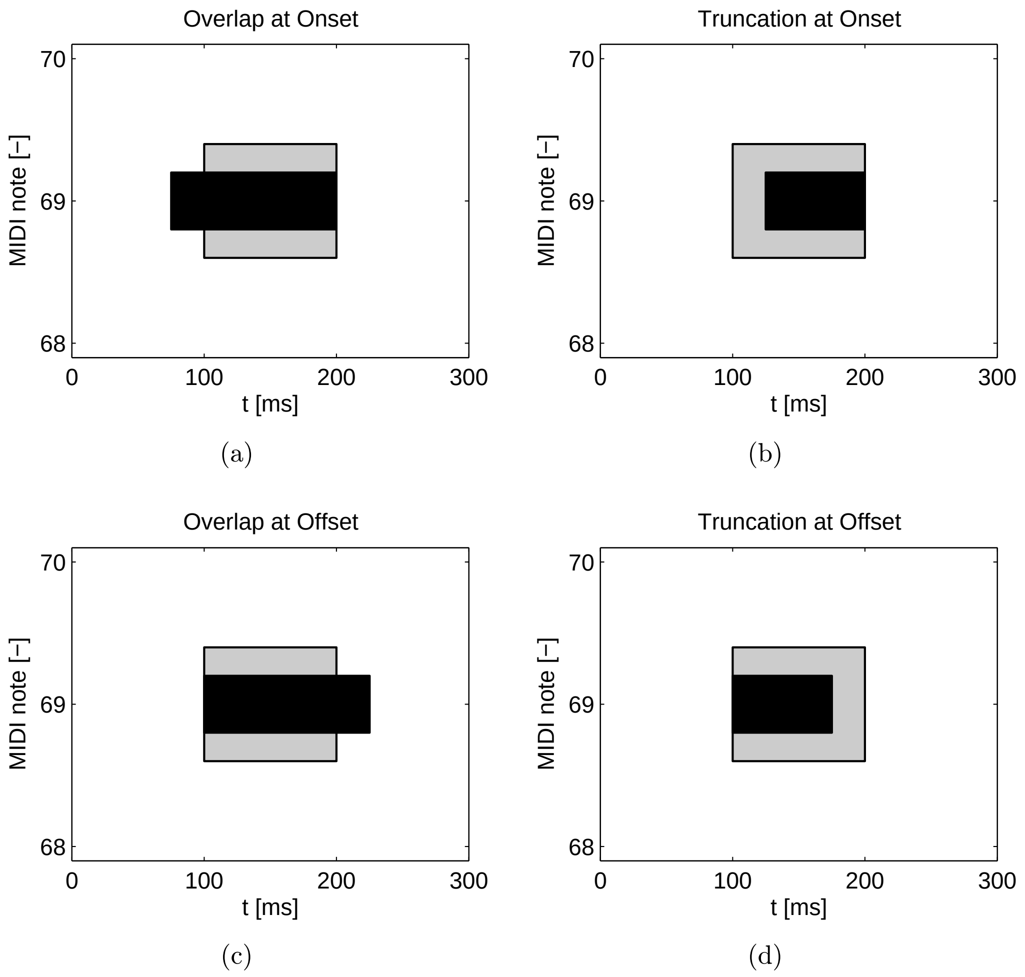


Figure 4.1: Illustration of time-based criteria

Since it is somewhat difficult to mathematically define the time-based criteria, Figure 4.1 provides an illustrative example. As can be seen, each "subfigure" shows the piano roll representation (see Section 2.3) of the *reference note* (painted gray) and the corresponding *transcribed note* (painted black). In all four cases, the frequency of both notes is 440 Hz corresponding to the MIDI note number 69 (note that transcribed notes are "narrower" only to be visually distinguishable from reference notes).

Figure 4.1(a) and 4.1(b) depict the *onset errors* OON and TON, respectively, which represent too early and too late detection of a note onset, whereas Figure 4.1(a) and 4.1(b) depict the *offset errors* OOF and TOF, respectively, which represent too late and too early detection of a note offset. Each criterion counts the number of samples between the reference and the detected event (i.e. onset or offset) and sums the contributions from all notes to obtain the total amount of a particular error in the transcription. Alternatively, this amount can be divided by the total duration of notes in the reference MIDI recording to obtain the proportional time error in percentage.

It should be emphasized that these criteria also add the contributions from the notes transcribed with a frequency error, provided that certain minimum time overlap is satisfied. The meaning of the overlap is clarified in the following section.

4.1.2 Overlapping measures

This section introduces two *overlapping measures* expressing the mutual time overlap within each *reference note - transcribed note pair*. When used in conjunction with correct pitch detection, the measures indicate the overall transcription correctness bilaterally in context of the reference and the transcription. Therefore, these measures are applied in Section 4.1.3 for classification of notes into specific categories constituting the note-based criteria.

Similarly as in the preceding section, definitions of both overlapping measures are given in words only, supported by examples displayed in Figure 4.2.

ROT - Reference Overlapping Transcription

This parameter indicates how much is the transcribed note covered by the reference note, i.e. how large portion of the transcribed note is indeed correct. If ROT is greater than 75%, for instance, it means that the transcribed note is correct in itself - although it may not entirely represent the reference note, it at least corresponds to some its certain portion, and thus increases the transcription quality. If $ROT = 100\%$, it merely means that the transcribed note is completely located inside the reference note (see Figure 4.2(c)), but does not indicate the transcribed area of the reference (which the purpose of the complementar measure TOR). It is interesting to mention that *ROT* penalizes both too long and too short transcriptions, as demonstrated in Figure 4.2(a) and 4.2(b), respectively.

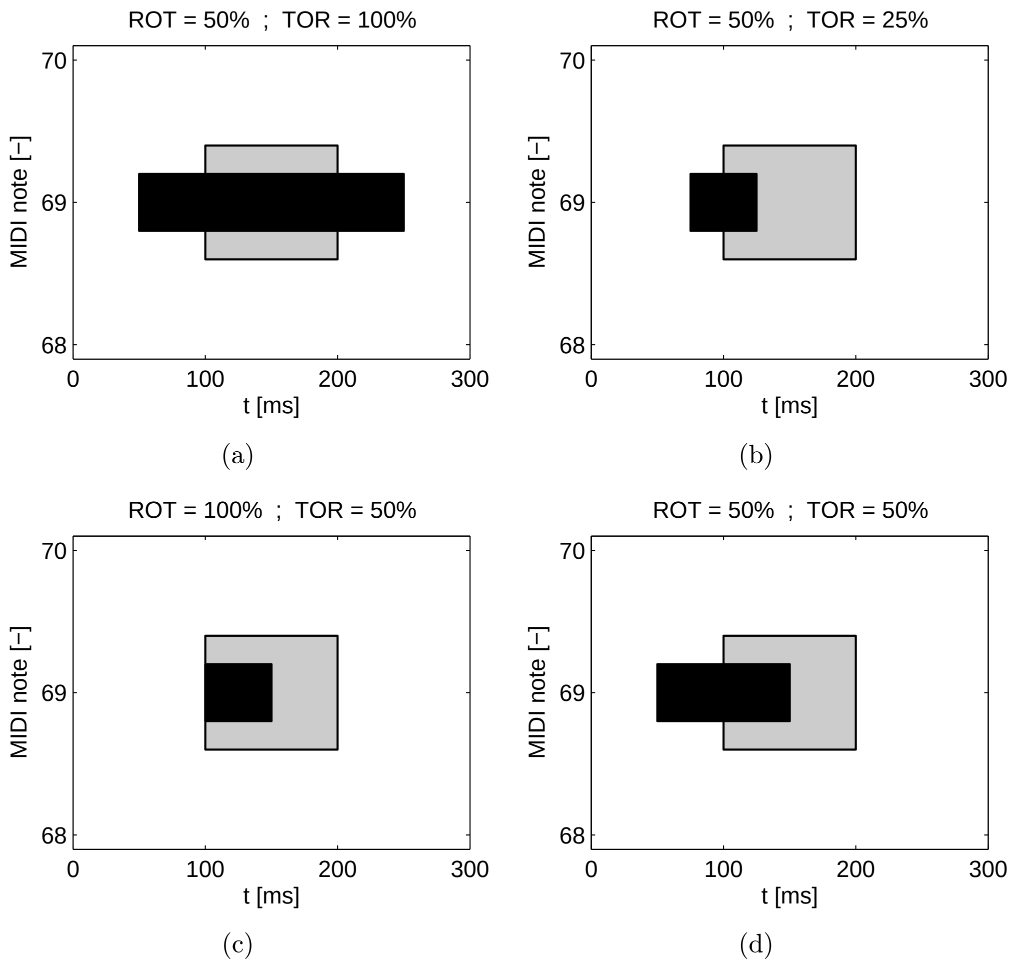


Figure 4.2: Overlapping measures between reference and transcribed notes

TOR - Transcription Overlapping Reference

This parameter indicates the percentage of the reference note covered by the transcribed note, i.e. how large portion of the reference is correctly detected. If $TOR = 100\%$, it only means that the transcribed note entirely overlaps the reference note, though the ideal result can be a subset of the transcribed note (see Figure 4.2(a)).

4.1.3 Note-based criteria

As the name implies, this section presents criteria performing the evaluation using the notes as independent units. As mentioned in Section 4.1, this approach share some ideas with [Ryynänen2004]. Nevertheless, our note-based evaluation is approached mainly from the reference notes perspective and compensates this imperfection by penalizing the transcriber for errors. On the other hand, the symmetry is in fact embedded in our conception as well, since the overlapping measures (see Section 4.1.2) characterize the bilateral relationship between the reference and transcribed notes.

The note-based evaluation classifies each reference-transcription pair into one of the *note categories*, according to the measures ROT and TOR, as well as the correctness of frequency detection. Each criterion then simply counts the number of notes in the respective note category defined in the sequel.

CTN - Completely Transcribed Notes

A reference note is *completely transcribed* by a note from the transcription, when the MIDI note frequencies agree and the notes exhibit large overlap in time:

$$f_{mid}^{ref} = f_{mid}^{trn} \tag{4.1}$$

$$ROT + TOR \geq 150\%, \qquad ROT \geq 60\%, \qquad TOR \geq 60\% \tag{4.2}$$

Three joint conditions in Equation (4.2) appear to be more flexible and yield better results than simple requirement of ROT or TOR to exceed a specific minimum value. Indeed, all four pairs in Figure 4.1 satisfy the above conditions, resulting in classification of the reference notes as CTN. On the other hand, the examples in Figure 4.2(a) and 4.2(c) violate the condition either for ROT or TOR (although satisfying the most difficult condition for the sum), and thus fall to the following category PTN.

PTN - Partially Transcribed Notes

A reference note is *partially transcribed* by a transcribed note when the frequency condition (4.1) is met and the notes satisfy less demanding overlap than CTN:

$$TOR + ROT \geq 100\%, \qquad TOR \geq 40\% \tag{4.3}$$

As suggested by an example PTN in Figure 4.2(d), this approach may result in a somewhat undervalued score, since listeners would probably consider the transcription as correct due to a hardly perceivable time shift. On the other hand, monophonic transcription is a significantly simpler task compared with polyphonic transcription, hence the quality demands should be much stricter. Moreover, such errors become considerably more audible with increasing note duration and decreasing tempo of the recording.

FER - Frequency ERrors

A reference note is classified as transcribed with a *frequency error*, when the notes exhibit time overlap defined in Equation (4.3) or (4.2), but Equation (4.1) is not fulfilled.

OER - Octave ERrors

Octave errors represent a special case of frequency error greater than or equal 12 semitones:

$$\left| f_{mid}^{\,ref} - f_{mid}^{\,trn} \right| \geq 12 \tag{4.4}$$

MIN - MIssed Notes

A reference note is classified as *missed* when no appropriate transcription candidate exists, or when the candidate is too inaccurate in time, regardless of the error in frequency detection. Specifically, the MIN criterion counts the references notes not identified by the previous criteria CTN, PTN, FER or OER.

FAN - FAlse Notes

This criterion counts the notes in the transcribed MIDI sequence not involved in the original recording. In other words, *false notes* are constituted by redundantly transcribed notes coupled with no reference note. In addition to that, a transcribed note is considered false (FAN) whenever the corresponding reference note is classified as missed (MIN). An illustrative example of this situation is depicted in Figure 4.2(b).

NDA - Note Detection Accuracy

Based on the preceding note criteria, we can characterize the overall quality of transcription by introducing the *Note Detection Accuracy* parameter defined as:

$$NDA = \frac{CTN + \frac{1}{2} \cdot PTN - 2 \cdot OER - FAN}{N} \cdot 100\% \qquad (4.5)$$

where N is the total number of reference notes. As can be observed, this parameters takes into an account both the reference and the transcription point of view. While the former is represented by the CTN, PTN and OER criteria, the latter is described by the FAN criterion. FER errors are not penalized for two reasons. First, classification of a particular note as FER is automatically reflected as a proportional decrease in CTN or PTN criterion. Second, FER errors express the correctness of onset/offset detection. On the other hand, octave errors OER are strongly penalized since they symbolize gross errors causing especially unpleasant impression of the transcribed melody.

4.2 Preparation of Testing Signals

This section provides brief description of the two methods used to prepare suitable test data. Although [Ryynänen2004] reports rapid growth of music databases, no standardized database was unfortunately available to us.

Synthesized signals As the name suggests, synthesized signals can be obtained by synthesizing a WAV file from a MIDI file. This is especially attractive because the exact *reference score* is readily available, and moreover, it can automatically be accomplished by a software *Wave Table synthesizer* (such as [TiMidity2004]) in order to prepare large database of testing signals. Unfortunately, the resulting WAV files generated by [TiMidity2004] contain undesirable amount of additive noise. Since filtering the noise would distort the audio signal, all reference WAV files were manually recorded using a sound card and [CoolEdit2004] during the playback of MIDI files in [Cakewalk2004].

Real signals Real signals from musical instruments were manually transcribed in order to obtain reference musical notation. First of all, note onsets and offsets were *labeled* similarly as in speech processing. Then, correct MIDI notes were detected by repeated listening, and the reference MIDI file was generated using the software by [Cemgil2004]. Finally, the resulting MIDI file was played several times to adjust the offsets in such a way, that all notes *sound* as closely as possible as the original instrument.

Title	Author	Instrument	Genre
Violin scale	unknown	violin	etude
Chameleon	Herbie Hancock	synthesizer	jazz
Horn fanfare	unknown	horn	brass music
Kadaň City Blues	Natrávená 5	saxophone	blues
Could You Be Loved?	Bob Marley	MIDI guitar	reggae

Table 4.2: Description of the testing audio signals

Since the number of testing signals is small, we have chosen musical instruments with very distinctive frequency spectra, as well as recordings from various musical styles. The testing signals are summarized in Table 4.2

4.3 Settings of the Transcription System

This section presents the parameters settings of the proposed transcription system. Since the desired task of the system is the *automatic* transcription of music, all parameters retain constant values shown in Table 4.3. The only exception is the *number of harmonics* parameter, which strongly affects the results of the pitch detection algorithm (see Section 3.1.4) and must be therefore individually tuned for each musical instrument. The chosen values are given in Table 4.4.

4.4 Transcription Examples

This section presents several examples of transcribed audio signals and discusses the results based on the criteria presented in Section 4.1. It is intended as a study of the transcription system properties, rather than generalization based on statistical evaluation. Transcription results are commented in words for each instrument, and summarized in Table 4.4.

4.4.1 Violin

This signal was used in Section 3.1.2 for demonstration of efficient spectral analysis with the constant Q transform, as depicted on Figure 3.4 and 3.5(a).[1] For this reason, the transcription results are almost perfect using only the pitch detection algorithm, as shown in Figure 3.8(a). Nevertheless, the onset

[1]These figures are similar to [Brown1991, Fig. 5]. The audio recording can be obtained at [Brown2004].

Parameter	Value
window type	Hamming
window length	93 ms
overlapping	85%
number of bins per octave, b	24
maximum frequency, f_{max}	$f_s/2$
minimum frequency, f_{min}	16.35 Hz
threshold for CQT matrix, *minval*	0.054
DFT phase-based correction	off
AR model order, p	30
sliding window length, L	40 ms
threshold in Hinkley's test, λ	40
bias in Hinkley's test, δ	0.2
audibility threshold, P_{aud}	0.05
time constant of leaky integrator, τ	93 ms
window length of moving average	30 ms

Table 4.3: Default algorithm parameters

Instrument	harm [-]	notes [-]	avgdur [ms]	NDA [%]	OON [%]	TON [%]	OOF [%]	TOF [%]
violin	14	15	156	100	2.98	0.67	0.67	2.94
synthesizer	6	12	257	87.5	0.20	5.17	3.38	0.07
horn	4	23	120	78.3	8.71	5.56	10.11	6.32
saxophone	14	28	193	76.8	7.50	3.89	4.78	16.35
MIDI guitar	3	10	169	70	3.01	13.52	58.14	0

Table 4.4: Summary of transcription results (*harm* - number of harmonics parameter, *notes* - total number of notes in the recording, *avgdur* - average duration of notes)

detector further refines the signal segmentation, and heuristic rules eliminate a short false note shown on Figure 3.16(b). As a result, the transcription is errorless in terms of note-based criteria, and the time error is negligible despite the rapidity of the recording.

4.4.2 Synthesizer

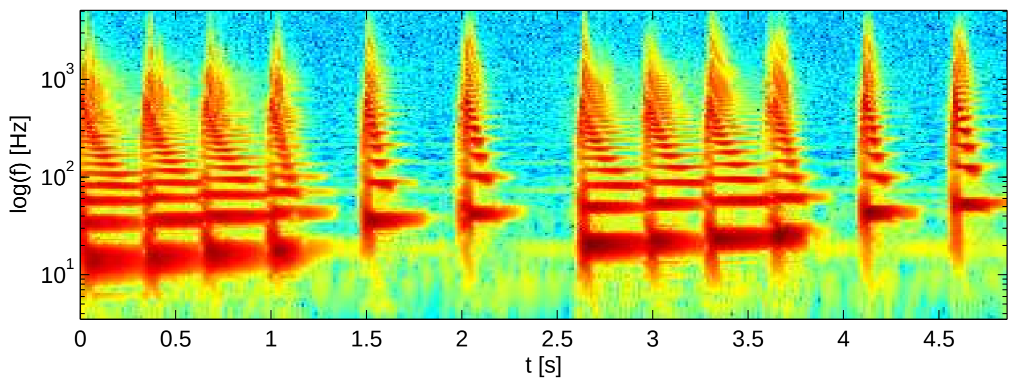
This example shows the ability of the transcription system to deal with very low-pitched instruments (around 50 Hz). Although the audio signal comes from a CD recording, the instrument itself is a synthesizer imitating the real bass. Therefore, the CQT spectrum exhibits strong fundamental frequency (see Figure 4.3(a)) responsible for relatively successful transcription (NDA = 87,5%). As can be seen in Figure 4.3(b), one note was detected partially, one false note was generated, and all remaining notes were transcribed correctly.

The time criteria yield the following results: TON = 5.2%, OOF = 3.4%, OON = 0.2%, TOF = 0.07%. This means that note onsets are predominantly detected with a slight delay originating in the Hinkley's test (see Section 3.2.2), and only very seldom the detection occurs before the actual onset. Similar situation happens with offsets, delayed due to detection based on simple thresholding of signal power. Histograms of TON and OOF criteria are shown on Figure 4.3(c) and 4.3(d), respectively. As can be observed, majority of the former error is caused by a single note (classified as PTN), whereas the latter error is formed by comparable constributions of almost all notes.

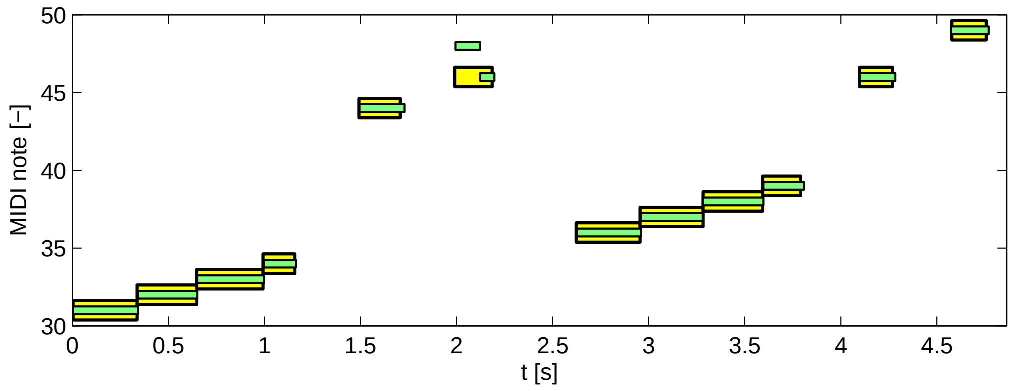
4.4.3 Horn

This example of horn melody illustrates a typical drawback of the transcription system. As can be observed in Figure 4.4(a), the onset detection algorithm fails several times to detect an evident note onset, resulting in three missed notes. This error can be reduced by decreasing the threshold λ to accept smaller "jumps" of the Kullback's divergence corresponding to less abrupt changes in the audio signal. However, this may introduce undesired oversegmentation causing division of longer notes into several shorter fragments.

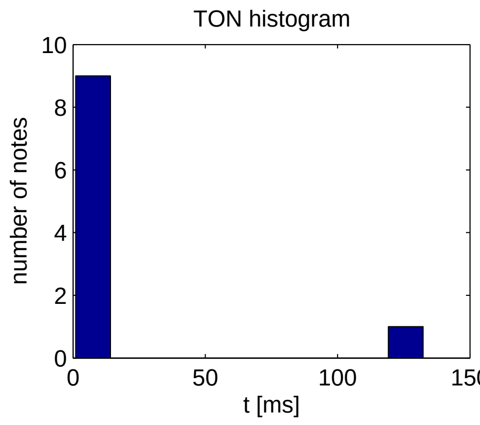
As can be observed in Figure 4.4(c), both onsets and offsets tend to be overlapped rather than truncated, though the segmentation is excellent for majority of notes. As shown in Figure 4.4(b) and 4.4(d), the frequency was incorrectly detected only in a single case, resulting in satisfactory transcription with NDA = 78,3%.

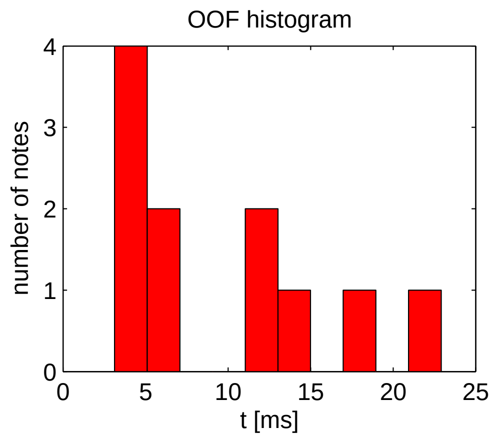


(a)



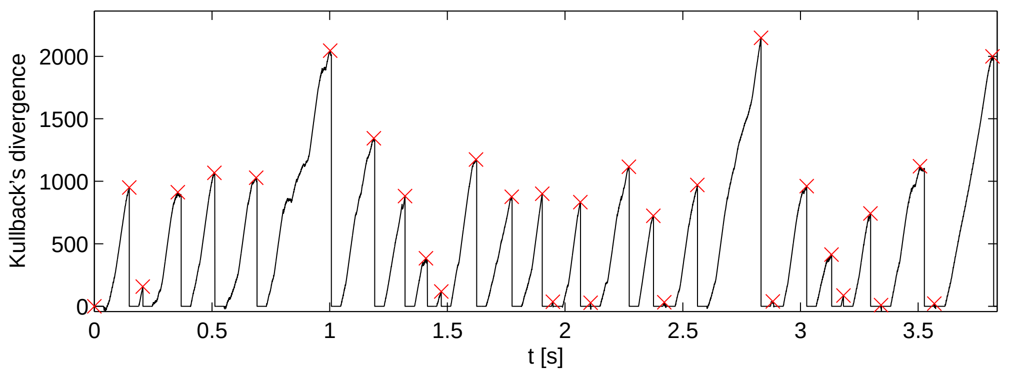
(b)



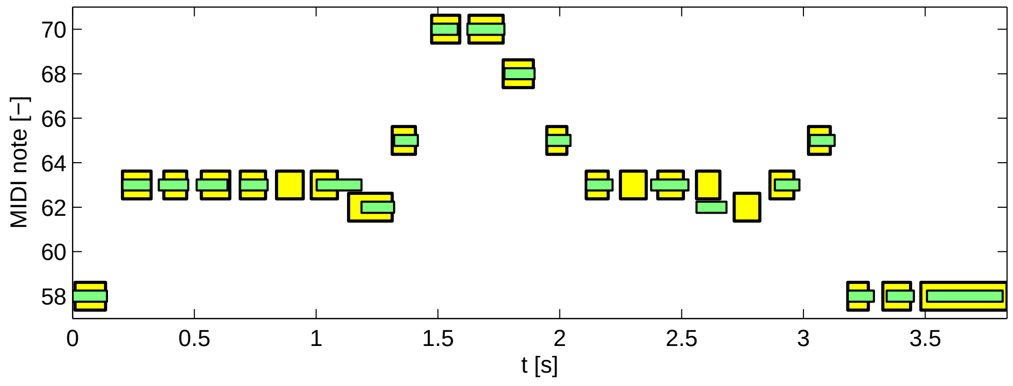


(c) (d)

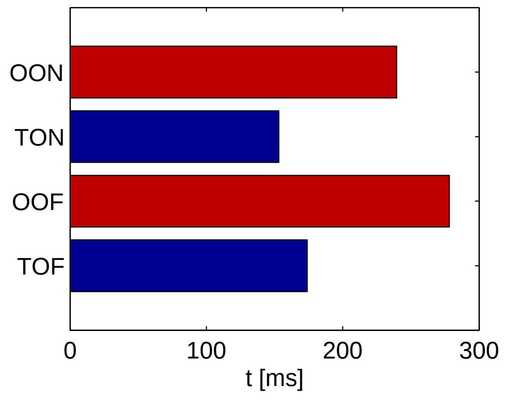
Figure 4.3: Transcription of synthesizer: (a) CQT spectrogram, (b) Reference melody (yellow) and transcription (green), Histogram of note contributions to the time error TON (c) and OOF (d)



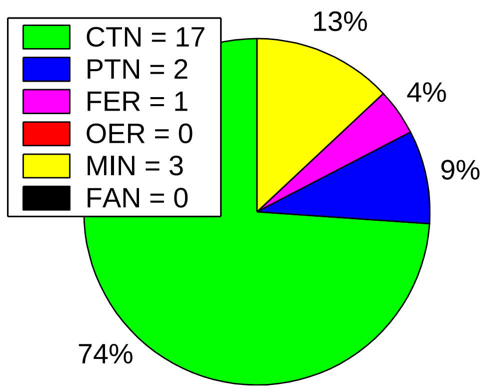
(a)



(b)



(c)



(d)

Figure 4.4: Transcription of horn: (a) Distance measure in the divergence test, (b) Reference melody (yellow) and transcription (green), (c) Time-based criteria, (d) Note-based criteria

4.4.4 Saxophone

This recording[2] presents a piece of blues solo played on alto saxophone. Contrary to pure sound of horn, this wind instrument is characterized by harmonically rich spectrum containing approximately 14 harmonics (see Table 4.4). As can be seen in Figure 4.5(d), the main shortcoming of the transcription are the frequency errors caused by the *Combining the Results* block. Indeed, the comparison of the note at t = 5 s in Figure 4.5(a) and 4.5(b) reveals that the fundamental frequency was detected correctly, but the heuristic rules assigned wrong frequency to the note candidate. The frequency error originates in spectral analysis of the note attack, which often yields uncertain frequency estimates due to noisy character of the sound. Consequently, the decision procedure rejected the true frequency candidate owing to specific conditions in signal power.

It seems that this problem could be easily solved by simple requirement of certain *minimum note duration*. However, this may eliminate correct notes in rapid recordings (typical for jazz or hard rock), because some false notes can last for almost 100 ms.

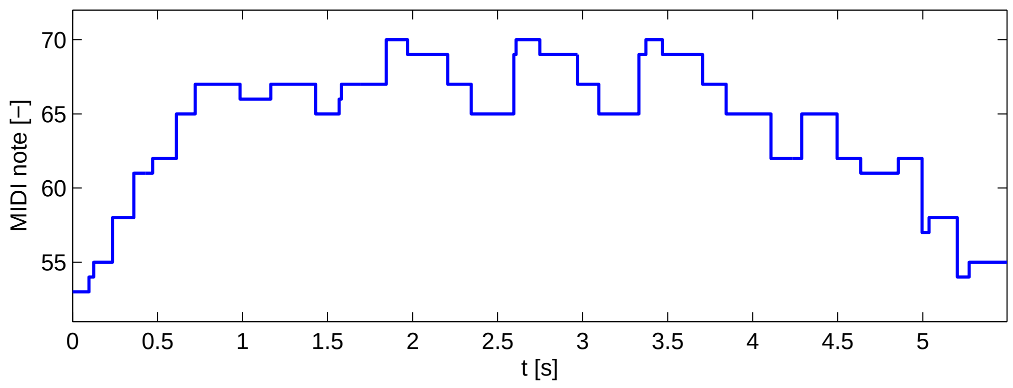
4.4.5 MIDI Guitar

This signal was obtained by MIDI to WAV conversion described in Section 4.2. As can be observed in Figure 4.6(a), the signal waveform is visibly artificial, which may imply simplicity of the transcription task. However, the guitar melody consists of multiple short notes of the same frequency,[3] and is therefore difficult for the segmentation algorithm. In fact, it would be hardly possible to resolve these notes without the onset detector, since the pitch tracking algorithm only detects a frequency "line" of long duration.

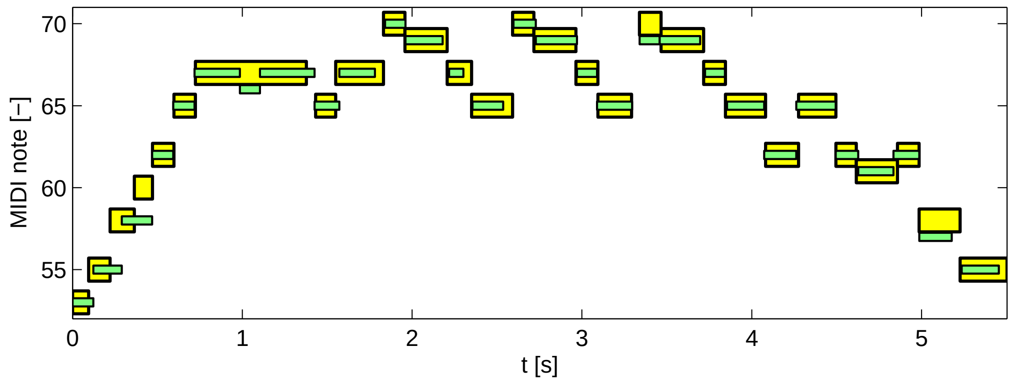
As can be seen in Figure 4.6(d), the overlap at offset error is extremely high (OOF = 58%), although the transcription sounds similarly as the reference. This can be explained by comparing the signal envelope with the reference MIDI sequence in Figure 4.6(a) and 4.6(b) respectively, which reveals that each correct MIDI offset is located approximately in the middle of the *release* region (recall the ADSR envelope model in Section 3.3). As a consequence, it is clearly impossible to precisely detect such offsets since the signal power is too high. For this reason, the demands on offset detection accuracy should probably be less restrictive.

[2]Inclusion of this song expresses author's pride and joy of his hometown, as well as his former band.

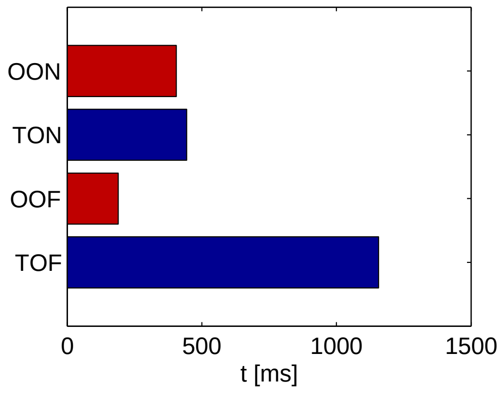
[3]This is a characteristic feature of bass and lead guitar melodies in Jamaican music, such as reggae, ska and dub. Another typical example is shown in Figure 3.8(b).



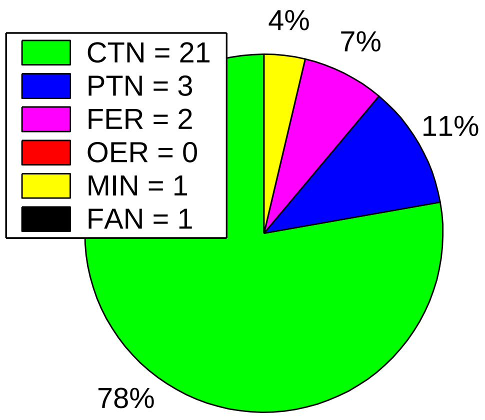
(a)



(b)

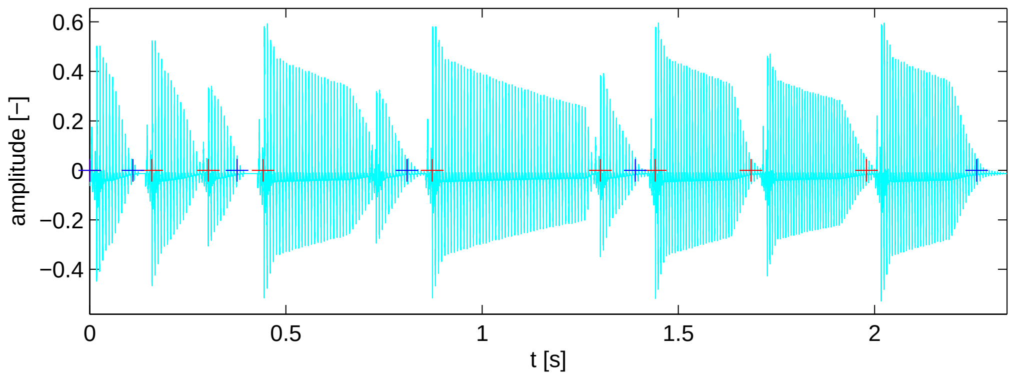


(c)

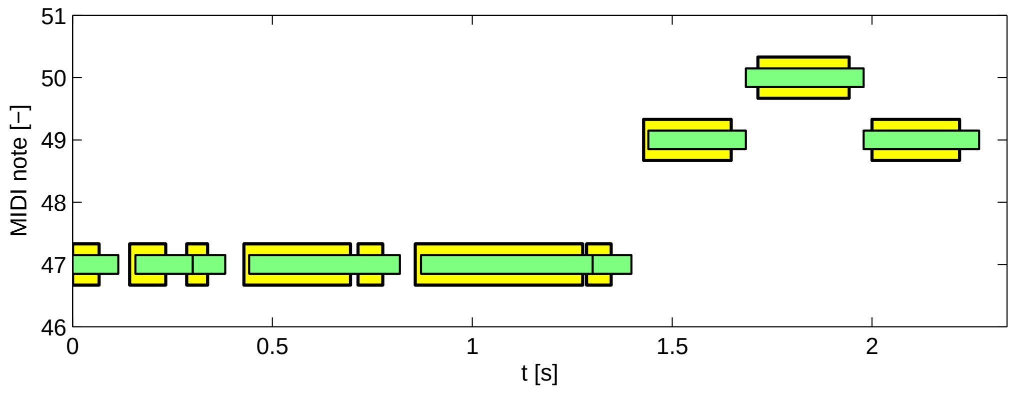


(d)

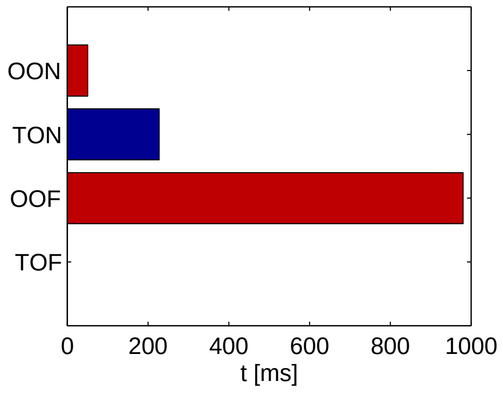
Figure 4.5: Transcription of saxophone: (a) Frequency tracking , (b) Reference melody (yellow) and transcription (green), (c) Time-based criteria, (d) Note-based criteria



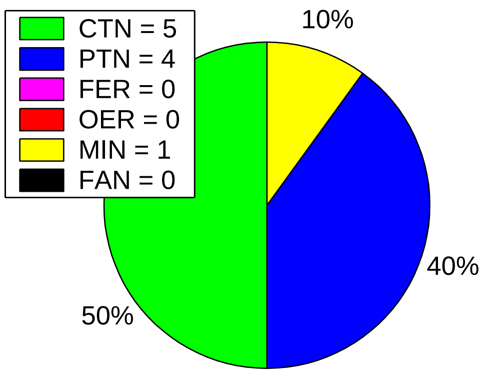
(a)



(b)



(c)



(d)

Figure 4.6: Transcription of MIDI guitar: (a) Signal waveform, (b) Reference melody (yellow) and transcription (green), (c) Time-based criteria, (d) Note-based criteria

Chapter 5

Conclusions

This thesis is concerned with automatic transcription of monophonic audio signals to MIDI representation. In the initial stage, large documentation research was done in order to find a solution which would be modern and robust, yet manageable by an undergraduate student. I am happy to state that my decision was correct, and I have finally managed to develop a transcription system which is modular, well documented, and capable of application in real enviroment.

An essential part of every music transcription system is a reliable fundamental frequency detector. I have chosen the solution based on the Constant Q Transform (CQT) for two main reasons. First, the Matlab implementation was available by the author [Brown1991], which greatly simplified the implementation. Second, recent publications in this field [Cemgil1995a], [Blankertz2004] referenced CQT as a suitable and efficient method for variety of signals. Therefore, I have studied and optimized the code for calculation of CQT, and subsequently implemented the rest of the pitch detection algorithm shown in Figure 3.6.

Although the pitch detector in itself is able to transcribe some signals, equal importance in the system is given to the onset detector, which is responsible for precise signal segmentation. Partially implemented solution was again found in the literature [Jehan1997], though it resembled a "programmer's nightmare" to rewrite the algorithm exactly as published by [Basseville1983]. Great advantage of this algorithm is the statistical time-domain approach performed on a sample-by-sample basis, which helped to improve the time resolution of CQT, constrained by the time-frequency tradeoff.

The design of the transcription system was accomplished by development of several heuristic rules which combine the results of both algorithms, and extract the final onset times and MIDI note frequencies. Then, an emphasis was put on development of graphical user interface (GUI) to provide an effective tool to analyze and evaluate the transcription process in a single environment. Such tool can be utilized in future with other pitch or onset detection algorithms, and moreover, it can also include a separate algorithm for instrument recognition, which would be an interesting future task.

Finally, new criteria for the transcription performance evaluation were developed. Although not proved by extensive statistical simulations, the method appears to be suitable for rapid musical passages, able to deal with various musical sounds, and applicable within a wide range of MIDI frequencies.

I regret that the article [Brown1993] is finally not functional in the transcription system. It was attempted to implement, but unfortunately, the final results yielded no improvement probably due to lack of my understanding to problems of phase unwrapping. Therefore, it could be a future task, as well as the above mentioned statistical evaluation on large signal database.

Bibliography

[Andre-Obrecht1986] Andre-Obrecht, R.: *A New Statistical Approach for the Automatic Segmentation of Continuous Speech Signals.* INRIA Research Report, RR-0511, 38 pages, March 1986 `<http://www.inria.fr/rrrt/rr-0511.html>`

[Andre-Obrecht1988] Andre-Obrecht, R.: *A New Statistical Approach for the Automatic Segmentation of Continuous Speech Signals.* IEEE Trans. on Acoust., Speech, and Signal Processing, Vol. 36, No. 1, Jan. 1988

[Basseville1982] Basseville, M. - Benveniste, A.: *Detection sequentielle de changements brusques des caracteristiques spectrales d'un signal numerique* (in English). INRIA Research Report, RR-0129, 89 pages, April 1982 `<http://www.inria.fr/rrrt/rr-0129.html>`

[Basseville1983] Basseville, M. - Benveniste, A.: *Sequential Detection of Abrupt Changes in Spectral Characteristics of Digital Signals.* IEEE Trans. on Information Theory, Vol. 29, No. 5, p. 709-724, Sep. 1983

[Basseville2003] Basseville, M. - Nikiforov I.V.: *Detection of Abrupt Changes - Theory and Application.* [online]. [cit. May 2004], `<http://www.irisa.fr/sigma2/kniga/>`

[Bello2002] Bello, J. P. - Monti, G. - Sandler, M. B.: *Automatic Music Transcription and Audio Source Separation.* Cybernetics and Systems: An International Journal, 33: p. 603-327, 2002

[Blankertz2004] Blankertz, B.: *The Constant Q Transform.* [online]. [cit. May 2004], `<http://wwwmath.uni-muenster.de/math/inst/logik/ /org/staff/blankertz/constQ/constQ.pdf>`

[Bořil2003] Bořil, H.: *Kytarový MIDI převodník* (in Czech). MSc. Thesis, Czech Technical University, 2003

[Brown1991] Brown, J.C.: *Calculation of a Constant Q Spectral Transform.* Journal of Acoustical Society America, Vol. 89, p. 425-434, Jan. 1991

[Brown1992a] Brown, J.C.: *Musical Fundamental Frequency Tracking using a Pattern Recognition Method.* Journal of Acoustical Society America, Vol. 92, p. 1394-1402, Sep. 1992

[Brown1992b] Brown, J.C.: *An Efficient Algorithm for the Calculation of a Constant Q Transform.* Journal of Acoustical Society America, Vol. 92, p. 2698-2701, Nov. 1992

[Brown1993] Brown, J.C.: *A High Resolution Fundamental Frequency Determination Based on Phase Changes of the Fourier Transform.* Journal of Acoustical Society America, Vol. 94, No. 2, p. 662-667, Aug. 1993

[Brown2004] Brown, J.C.: *Constant Q Transform* [online]. [cit. May 2004], `<http://web.media.mit.edu/~brown/cqtrans.htm>`

[Cakewalk2004] Twelve Tone Systems, Inc: *Cakewalk* [online]. [cit. May 2004], `<http://www.cakewalk.com/>`

[Cemgil1995a] Cemgil, A.T.: *Automated Monophonic Music Transcription (A Wavelet Theoretical Approach).* MSc. Thesis, Bogazici University, 1995, [online]. `<http://carol.science.uva.nl/~cemgil/papers/tez.zip>`

[Cemgil1995b] Cemgil, A.T. - Anarim, E. - Caglar, H.: *Comparison of Wavelet Filters for Pitch Detection of Monophonic Music Signals.* European Conference on Circuit Theory And Design, Vol. 2, p. 711-14, 1995

[Cemgil2004] Cemgil, A.T.: *Software and Utilities* [online]. [cit. May 2004], `<http://carol.science.uva.nl/~cemgil/software.html>`

[CoolEdit2004] Syntrillium Software Corporation: *Cool Edit 2000* [online]. [cit. May 2004], `<http://www.syntrillium.com/cooledit>`

[Glatt2004] Glatt, J.: *MIDI is the language of gods* [online]. [cit. May 2004], `<http://www.borg.com/~jglatt/>`

[Hess1983] Hess, W.: *Pitch Determination of Speech Signals.* Springer-Verlag, Berlin, 1983

[Jehan1997] Jehan, T.: *Musical Signal Parameter Estimation.* Msc. Thesis, CNMAT, Berkeley, 1997, [online]. [cit. May 2004], `<http://www.cnmat.berkeley.edu/~tristan/Thesis/timedomain.html>`

[Klapuri1998] Klapuri A.: *Automatic Transcription of Music.* MSc. Thesis, Tampere University of Technology, April 1998, [online]. [cit. May 2004], `<http://www.cs.tut.fi/sgn/arg/music/klapuri2.pdf>`

[Martins2001] Martins, L.G.: *PCM to Midi Transposition.* MSc. Thesis, Universidade do Porto, 2001, [online]. [cit. May 2004], `<http://telecom.inescn.pt/research/audio/p2m/>`

[Medan1991] Medan Y., Yair E., Chazan, D.: *Super Resolution Pitch Determination of Speech Signals.* IEEE Trans. on Signal Processing, Vol. 39, No. 1, Jan. 1991

[MMA2004] MIDI Manufacturer's Association: *Official webpage* [online]. [cit. May 2004], `<http://www.midi.org/>`

[Pollák2002] Pollák P.: *Criteria for VAD Classification.* Internal Research Report R02-1, Czech Technical University in Prague, Dec. 2002

[Rabiner1975] Rabiner, L.R. - Sambur, M.R. - Schmidt, C.E.: *Applications of a Nonlinear Smoothing Algorithm to Speech Processing.* IEEE Trans. on Acoust., Speech, and Signal Processing, Vol. ASSP-23, No. 6, Dec. 1975

[Rabiner1976] Rabiner, L.R. et. al: *Comparative Performance Study of Several Pitch Detection Algorithms.* IEEE Trans. on Acoust., Speech, and Signal Processing, vol. ASSP-24, No. 5 Dec. 1976

[Rosca2002] Rosca J. - Balan R. - Fan N.P. - Beaugeant C. - Gilg V.: *Multichannel Voice Detection in Adverse Enviroments.* In Proc of EUSIPCO 2002, Toulouse, France, Sep. 2002

[Ryynänen2004] Ryynänen M.: *Probabilistic Modelling of Note Events in the Transcription of Monophonic Melodies.* MSc. Thesis, Tampere University of Technology, March 2004, [online]. [cit. May 2004],
`<http://www.cs.tut.fi/sgn/arg/matti/mryynane_thesis.pdf>`

[Sovka2003] Sovka, P. - Pollák, P., *Vybrané metody číslicového zpracování signálů* (in Czech). Vydavatelství ČVUT, Praha, 2003, 2.vydání

[TiMidity2004] Tim Brechbill: *TiMidity++ Download Site* [online]. [cit. May 2004], `<http://onefreehost.com/saxguru/Timidity.html>`

[Youngberg1979] Youngberg, J.E.: *Rate/Pitch Modification Using the Constant-Q Transform.* In Proc. ICASSP 79, p. 748-51, April 1979

Appendix A

List of Source Codes

`wav2mid.m`	Main function of the transcription system

A.1 Pitch Detection

`CQTmtx.m`	CQT matrix (temporal and spectral kernels)
`CQTpattern.m`	Ideal pattern of spacing of harmonic frequency components
`f2mid.m`	MIDI note quantization or interpretation
`phiCorrect.m`	Pitch detection based on phase changes of DFT
`sigMat.m`	Reshape signal into a matrix form
`stcqt.m`	Constant Q Transform of a signal

A.2 Detection of Events

`divTest.m`	Interface function with `mxDivTest.dll`
`mxDivTest.dll`	Matlab MEX-file for computation of the divergence test
`mxDivTest.c`	Source code for compilation of `mxDivTest.dll`
`divTestC.m`	Interface with the C implementation of the divergence test
`divTestM.m`	Divergence Test algorithm - Matlab implementation
`recurBurg.m`	Identification of the global autoregressive model M_0
`acorrAR.m`	Identification of the local autoregressive model M_1
`recurACF.m`	Recursive computation of the autocorrelation function
`chshape.m`	Matrix reorganization for multichannel signals
`d2int.m`	Conversion of a signal from double to integer (for `savebin.m`)
`savebin.m`	Save vector as integer binary data file

	Levinson recursion (reflection coefficients → AR coefficients)
`rc2ar.m`	- Matlab help file
`rc2ar.c`	- C source code
`rc2ar.dll`	- Matlab MEX-file
	AR coefficients → cepstral coefficients
`ar2cc0.m`	- Matlab help file
`ar2cc.c`	- C source code
`ar2cc.dll`	- Matlab MEX-file
	Six modifications of cepstral distance
`cepDist.m`	- Matlab help file
`cepDist.c`	- C source code
`cepDist.dll`	- Matlab MEX-file

A.3 Estimation of Power

`EstPow.m`	Block and recursive estimate of short-time power
`localExtr.m`	Find local extremes to a given level of significance

A.4 Combining the Results

`combResults.m`	Combine the results using heuristic rules
`saveRes.m`	Auxiliary function for saving decision results

A.5 Notes to MIDI Conversion

A.5.1 Notetrack Methods

`display.m`	Display Function
`horzcat.m`	Merges two notetracks
`length.m`	Number of Note events
`minus.m`	Shifts Note by backward by delta time
`mpower.m`	Transpose the pitches by halfnote steps
`mrdivide.m`	Scales durations and onset times by factor
`mtimes.m`	Scales durations and onset times by factor
`notetrack.m`	Constructor (load)
`save.m`	Saves as a midi File
`subsasgn.m`	Subscripted assignment
`subsref.m`	Subscripted Reference to Notetrack Objects
`play.m`	Plays a notetrack (needs an external program)
`plot.m`	Plot Function
`plus.m`	Shifts Note forward by delta time
`vertcat.m`	Concatenation of two tracks in time

A.5.2 C++ Source Codes

`common.cpp`	Support classes
`m_ifstrm.cpp`	Midi Input Stream
`m_ofstrm.cpp`	Midi Output Stream
`midisong.cpp`	Container Classes
`mid2txt.cpp`	Test Program
`midi2notes.cpp`	Mex function to read a MIDI file into a cell array
`notes2midi.cpp`	Mex function to write a cell array into a midi file

A.6 Evaluation of Transcription Quality

`createRefMid.m`	Create reference MIDI file given a labeled information
`cellToStruct.m`	Convert cell array to structure array
`compareMid.m`	Compare reference and transcribed midi file for criteria evaluation
`evalCrit.m`	Evaluate music transcription criteria
`getNoteTrack.m`	Return notetrack objects for reference and transcription midi file
`testCritEval.m`	Script for time-based and note-based criteria evaluation

A.7 WAV to MIDI Tool (GUI)

`wav2midTool.m`	Main GUI window for transcription, playback and displaying
`paramEdit.m`	GUI window for adjusting and loading or saving parameters
`paramEditIni.m`	Initialization script for `paramedit.m`
`wav2midTest.m`	Script for executing the algorithm without the GUI
`loadSig.m`	Load the signal to be transcribed by `wav2midtest.m`
`defaultParam.m`	Return default parameters for the transcription system
`compAuxParam.m`	Calculate auxiliary parameters (for showing in GUI)
`copyAxes.m`	Create a copy of current axes in a new figure
`mid2wav.m`	Synthesis of a wav file from a midi file
`pianoroll.m`	Piano roll plot of a midi file