

Insert here your thesis' task.



**FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE**

Master's thesis

Multi-instrument music transcription

Bc. Yevhen Kuzmovych

Department of Applied Mathematics

Supervisor: Ing. Marek Šmíd, Ph.D.

January 9, 2020

Acknowledgements

I would like to thank the supervisor of the work, Ing. Marek Šmíd, Ph.D. for the help in writing of this thesis, valuable advice and suggestions for improvement, thorough reviews, and extensive knowledge sharing in the fields of signal processing and ML.

I want to express my special gratitude to my family, especially my parents, for their great support and help throughout the whole study and writing of this work¹.

¹Я бы хотел выразить особую благодарность моей семье, особенно моим родителям, за их большую поддержку и помощь на протяжении всего обучения и написания этой работы.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on January 9, 2020

.....

Czech Technical University in Prague

Faculty of Information Technology

© 2020 Yevhen Kuzmovich. All rights reserved.

This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis

Kuzmovich, Yevhen. *Multi-instrument music transcription*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

Abstrakt

Přepis hudby hraje důležitou roli v oblasti získávání hudebních informací, a je to často složitý úkol i pro lidského posluchače. Automatizace tohoto přepisu se těší velkému zájmu z řad studentů hudby, kteří jej mohou využít pro nácvik rozpoznávání not a učení nových skladeb. Také hudební producenti a DJ, kteří by mohli využít funkce detekce stupnice a oddělení zdrojových nástrojů, by z něj mohli těžit. Podobně streamingové platformy ho mohou používat pro své doporučovací systémy, atp.

Tato práce zkoumá nejmodernější řešení, která kombinují analýzu signálů s přístupy strojového učení s učitelem pro uvedené problémy. Navrhuje implementaci, která je využita k provádění úlohy transkripce hudby s více nástroji, která má na vstupu surový zvukový soubor a produkuje sadu notových listů pro každý použitý nástroj na výstupu. Řešení je implementováno v programovacím jazyce Python jako modul a současně jako aplikace s rozhraním pro příkazovou řádku. Implementace je rozdělena do logických modulů zodpovědných za odhad odpovídajících částí výstupní partitury, jmenovitě oddělení zdrojů, detekce výšek a událostí, odhad tempa, stupnice a taktového předznamení. Takové oddělení umožňuje jednoduché rozšiřování a testování. Přesnost každého modulu je vyhodnocena na příslušných datových sadách.

Klíčová slova Přepis hudby, zpracování zvuku, analýza signálu, učení s učitelem.

Abstract

Music transcription is important though complex, often even for a human listener, task in the field of music information retrieval. Automation of such task is in a big demand among musical students, who can use it for practicing of note recognition and learning of the new pieces; music producers and DJs, who could utilize its key detection and source instrument separation functionality; streaming platforms that may use it for their recommendation systems; etc.

This thesis explores state-of-the-art solutions that combine signal analysis and supervised machine learning approaches for the mentioned problems. It proposes implementation that utilizes them to perform a multiple-instrument music transcription task having a raw audio file on the input and producing a set of sheet music scores for each played instrument on the output. The solution is implemented in the Python programming language as a module as well as a command-line interface application. The implementation is separated into logical modules responsible for estimation of the corresponding parts of the output score, namely source separation, pitch and event detection, tempo, key and time signature estimation. Such separation allows for simple expansion and testing. The performance of each module is evaluated on appropriate datasets.

Keywords Music transcription, audio processing, signal analysis, supervised learning.

Contents

Introduction	1
Problem definition	2
1 State-of-the-art	5
1.1 Source separation	5
1.2 Multi-pitch Detection	7
1.3 Note Tracking	10
1.4 Tuning, time signature, key, and tempo estimation	12
2 Analysis and design	15
2.1 Architecture	15
2.2 Audio streaming	17
2.3 Music source separation	18
2.4 Pitch extraction	18
2.5 Event detection	20
2.6 Tuning classification	22
2.7 Tempo estimation	23
2.8 Time signature estimation	23
2.9 Key classification	25
2.10 Post processing	26
2.11 Score generation	27
3 Implementation	29
3.1 Used tools	29
3.2 Project structure	29
4 Testing	33
4.1 Source separation	33
4.2 Sound envelope detection	34

Conclusion	37
Possible improvements	37
Bibliography	39
A Acronyms	45
B Musical notation	47
B.1 The Staff	48
B.2 Leger Lines	48
B.3 Clefs	48
B.4 Rhythmic Description	51
C Contents of enclosed CD	53

List of Figures

1.1	U-net's network architecture [3]	6
2.1	Architecture of the implementation.	16
2.2	Sound envelope.	19
2.3	Sound envelopes of piano and violin [44].	19
2.4	Data flow for pitch detection.	21
2.5	Base <i>A</i> commonly tuned to 440 Hz.	22
2.6	Notes in a key of <i>C major</i>	26
2.7	Notes in a key of <i>G major</i> (note a \sharp on the <i>F</i> line).	26
2.8	Rests detection algorithm.	27

List of Tables

2.1	Time signature selection table.	25
4.1	Spleeter and Open-Unmix performances.	34
4.2	Envelope detection models' performance.	34

Introduction

The problem of automated sound transcription of sheet music increases its popularity each year. The usage of a system capable to perform such task is very wide-ranging. It can be utilized by musical students to learn new musical peaces that they have only in digital format (in Waveform Audio File Format (WAVE) or MP3) or to train their pitch detection ability. The systems that are able to perform musical key detection (like *Mixed in Key*[1]) are commercially successful and are very popular among DJs and music producers who use them for mixing the music. And in general music information retrieving can bring a deeper insight into musical structure, quality, genre, etc. that can be used for many important real-world problems, including recommendation systems for streaming platforms.

The problem of music transcription might be complicated even for a human listener, may be simpler for a trained musician, but it appears very complex for the automated system. Since 1970, a whole research community has formed and has been improving the quality of automated transcription and reduced the constraints required for producing it.

But it is still not a resolved problem and there is no such system that is capable of automatically and accurately converting a recording of the musical piece back into sheet music or a set of commands that would replicate a recording on a music synthesizer.

This thesis proposes the implementation and its architecture that attempts to solve the required tasks to generate the correct transcription of the given musical piece. The architecture separates solutions for different problems into modules which allows for simple extension of the system and testing of individual approaches for the given problems. It also provides an estimated quality of each solution each evaluated on the appropriate datasets.

Problem definition

This thesis discusses the state-of-the-art solutions for the different problems that underlie a task of music transcription, namely: source instruments separation, pitch and event detection, musical key, tempo and time signature estimation that in combination determine necessary parts of the sheet music notation.

Short introduction to a sheet music notation is presented in Appendix B.

Pitch estimation

The difficulty of multi-pitch estimation lies in the fact that sound sources often overlap in time as well as in frequency. Hence, the extracted information may be partly ambiguous. Also, when musical notes are played in harmonic relations, the parts of higher notes may overlap completely with those of lower ones. Besides, spectral characteristics of musical instrument sounds, that define change of the note volume in time and timbre, are diverse, which increases the ambiguity in the estimation of amplitudes of sound sources. The resulting complexity leads not only to octave ambiguity but also the ambiguity in the estimation of the number of sources.

Tuning

Tuning, in general, is not a big problem for automated transcription systems. But it is important to note that recorded musical instruments may not be in tune or all the notes are tuned higher or lower but keeping the correct ratios. Estimation of a tuning, if used correctly, may improve the accuracy of the output score.

Another problem is that different tuning system can be used other than equal temperament, e.g. pythagorean tuning, meantone temperament, well temperament, etc. But as most of the Western music uses equal temperament system, only this system will be considered in the analysis.

Equal temperament is a tuning system in which the interval between each pair of consequent notes has the same ratio. Having chosen one base note and its frequency (commonly, *A* with 440 Hz is used), frequencies of all the other notes are defined to have $\sqrt[12]{2}$ ratio between adjacent notes, such that same note of a higher octave would have twice the frequency, as there are 12 semitones in a standard 12 tone system.

Key

Musical *key* is a part of sheet music notation. It defines a set of used notes and chords in a piece of music and their harmonic functions within the score. It simplifies the notation and understanding of the piece for a musician who reads it. And even though the detection of a set of used notes is a fairly simple

task, having notes detected within a pitch estimation, classification of key is still a big research field as it is not as straight-forward task. The same set of notes may indicate different keys depending on the context. For example, key of *C major* has the same set of notes as key of *A minor* and distinguishing between them is often not a simple task even for a human listener.

State-of-the-art

This chapter discusses existing state-of-the-art solutions for music transcription.

Sound transcription into sheet music is a combination of several techniques that include but are not limited to source instruments separation, pitch/note detection, event detection, etc.

Source separation and sound transcription to sheet music are fairly independent processes so their description and approaches may come from different sources and different projects. Therefore, implementation will also be separated.

1.1 Source separation

There were many successful attempts for music score source separation [2, 3, 4]. Performance of such projects are commonly measured according to *Source Separation campaign (SiSeC)* [5] on the standard *musdb18* [6] and *DSD100* [7] datasets.

Latest and most successful project in this field is *Spleeter* [2]. It is a project of Deezer². It takes similar approaches to previous solutions by University of London and Spotify [3]. Spleeter's pre-trained models will be used in the module responsible for music source separation described in detail in chapters 2 and 3.

Following approaches are described in [2, 3, 4].

1.1.1 Spleeter's approach

The pre-trained models are U-nets [3] and follow similar specifications as in *Singing voice separation: a study on training data* [4]. The U-net is an encoder/decoder Convolutional Neural Network (CNN) architecture with skip

²Deezer is a French online music streaming service (deezer.com).

connections [2]. Architecture used in this approach showed a state-of-the-art results on DSD100 dataset [3] and in the last SiSeC [7].

1.1.2 U-net architecture

The U-Net shares the same architecture (shown in Fig. 1.1) as a convolutional autoencoder with extra skip-connections that bring back detailed information lost during the encoding stage to the decoding stage. It has five strided³ 2D convolution layers in the encoder and five strided 2D deconvolution layers in the decoder.

The goal of the neural network architecture is to predict the vocal and instrumental components of its input indirectly: the output of the final decoder layer is a soft mask for each source that is multiplied element-wise with the mixed spectrogram to obtain the final estimate.

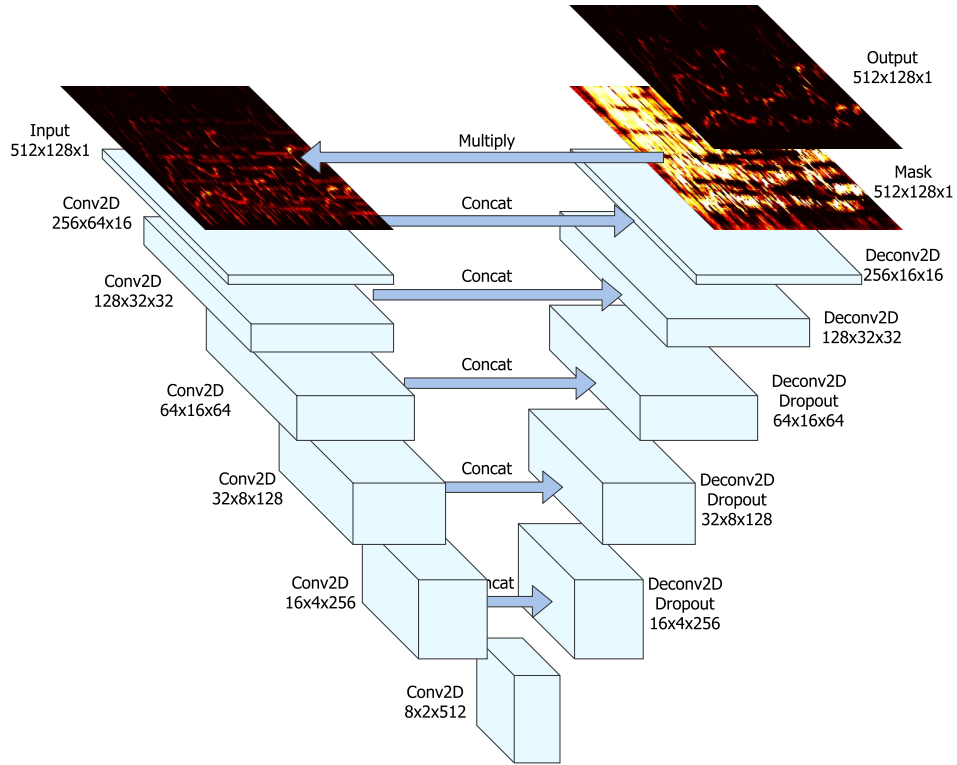


Figure 1.1: U-net’s network architecture [3]

³Transposed convolutions – also called *fractionally strided convolutions* – work by swapping the forward and backward passes of a convolution. One way to put it is to note that the kernel defines a convolution, but whether it’s a direct convolution or a transposed convolution is determined by how the forward and backward passes are computed [8].

1.1.3 Data and training

Spleeter’s training dataset is an internal Deezer’s dataset and is not shared (for copyright reasons).

Another project with similar approach, as explained in the dedicated article [4], uses two datasets during training of the models: *MUSDB* [6] and *Bean*(private dataset).

MUSDB is the largest and most up-to-date public dataset for source separation [6]. It contains 150 songs of western music genres primarily pop/rock, some hip-hop, rap and metal songs. And each song consists of four audio tracks: drums, bass, vocal and other. Original mix (and input of the model) is produced by summing tracks of four sources (expected outputs) together.

1.2 Multi-pitch Detection

There were several projects utilizing different approaches to a problem of Automatic music transcription (AMT). Following sections discuss these approaches and projects that used them.

The most important part of transcription of sound into sheet music is pitch (and subsequently note) detection. The core problem of polyphonic music transcription is multi-pitch detection.

In his Ph.D. research *Multiple fundamental frequency estimation of polyphonic recordings* [9], Chunghsin Yeh classifies multi-pitch detection systems according to their estimation type into two categories: joint and iterative. The *iterative* approach extracts the most eminent frequency per each iteration, until no other pitch can be estimated and extracted. Commonly, iterative estimators generate errors on each iteration but are much cheaper in terms of computation costs.

On the other side, the *joint* estimation models evaluate combinations of pitches at once, which leads to increase in accuracy but also in computation costs. Most of the latest approaches and state-of-the-art solutions fall into the joint category. Solution in this thesis also follows this category and will be discussed in detail in Chapter 2.

1.2.1 Feature-based multi-pitch detection

Most multi-pitch estimation and note tracking approaches exploit methods that come from signal processing. There is no specific model (Machine learning (ML) or other), and notes are detected using audio features that come from the input time-frequency representation (spectrogram) either in an iterative or joint way. Usually, multi-pitch estimation uses a *pitch candidate set score function* or a *pitch salience function*.

A *salience function* is a function that provides an estimation of the predominance of different frequencies in an audio signal at every time frame [10].

A *pitch candidate set score function* is a function designed to evaluate the plausibility of the combination of the hypothetical sources [9].

These feature-based techniques have produced the best results in the Music Information Retrieval Evaluation eXchange (MIREX) [11] multi-pitch and note tracking evaluations. The work by Chunghsin Yeh [9] was the best performing method in the MIREX multi-pitch and note tracking tasks. Yeh proposed a joint pitch estimation algorithm based on a pitch candidate set score function. Having a set of pitch candidates, the overlapping partials are detected and smoothed according to the spectral smoothness principle, which states that the spectral envelope⁴ of a musical sound tends to be slowly changing as a function of frequency. The score function for the pitch candidate set consists of four features: harmonicity, mean bandwidth, spectral centroid, and “synchronicity” (synchrony). A polyphony inference mechanism based on the score function increase selects the optimal pitch candidate set [9].

In the following year, the best performing method for the MIREX multi-pitch estimation and note tracking tasks, Karin Dressler described in her work *Multiple fundamental frequency extraction for MIREX* [13]. A multiresolution Fast Fourier transform (FFT) analysis was used as an input time/frequency representation, where the magnitude for each spectral bin was multiplied with the bin’s instantaneous frequency. Pitch estimation is made by identifying spectral peaks and performing pair-wise analysis on them, resulting on ranked peaks according to harmonicity, smoothness, the appearance of intermediate peaks, and harmonic number. Finally, the system tracks tones over time using an adaptive magnitude and a harmonic magnitude threshold.

Other notable feature-based AMT solution was introduced in the work by Pertusa and Inesta *Multiple fundamental frequency estimation using Gaussian smoothness and short context* [14]. They proposed a computationally inexpensive method for multi-pitch detection which computes a pitch salience function and evaluates combinations of pitch candidates using a measure of distance between a Harmonic partial sequence (HPS) and a smoothed HPS. Another approach for feature-based AMT was proposed in *Hybrid genetic algorithm based on gene fragment competition for polyphonic music transcription* [15], which uses genetic algorithms for estimating a transcription by mutating the solution until it matches a similarity criterion between the original signal and the synthesized transcribed signal.

More recently, Peter Grosche et al. proposed [16] an AMT method based on a mid-level representation derived from a multiresolution FFT combined with an instantaneous frequency estimation. His system also combines event (specifically start of the note) detection and tuning estimation for computing predictions. Finally, Juhan Nam et al. proposed [17] a classification-based approach for piano transcription using features learned from deep belief net-

⁴*Spectral envelope* of the sound determines the particular vowel sound produced, and is, in general, one of the important acoustic features that determine its perceived timbre [12].

works [18] for computing a mid-level time-pitch representation.

1.2.2 Statistical model-based multi-pitch detection

Many approaches in the literature formulate the multi-pitch estimation problem within a statistical framework. As Valentin Emiya et al. explains in their article *Multipitch Estimation of Piano Sounds Using a New Probabilistic Spectral Smoothness Principle* [19]: given an observed frame \mathbf{x} and a set \mathbf{C} of all possible fundamental frequency combinations, the frame-based multi-pitch estimation problem can then be viewed as a Maximum a posteriori (MAP) estimation problem:

$$\hat{C}_{MAP} = \arg \max_{C \in \mathbf{C}} P(C|\mathbf{x}) = \arg \max_{C \in \mathbf{C}} \frac{P(\mathbf{x}|C)P(C)}{P(\mathbf{x})}$$

where $C = \{F_0^1, \dots, F_0^N\}$ is a set of possible frequencies (considering tuning of an instrument), \mathbf{C} is the set of all possible F_0 combinations, and \mathbf{x} is the observed audio signal within a single analysis frame.

An example of MAP estimation-based transcription is the *PreFEst* system introduced by Masataka Goto in his article *A real-time music-scene-description system: predominant-F0 estimation for detecting melody and bass lines in real-world audio signals* [20], where each harmonic is modelled by a Gaussian centered at its position on the log-frequency axis. Expectation-maximisation (EM) algorithm is used to estimate MAP value. An extension of this method was proposed by Kameoka et al. in *A Multipitch Analyzer Based on Harmonic Temporal Structured Clustering* [21], which jointly estimates multiple possible frequencies, moments of start and end of the note, and dynamics. Partial harmonics are modelled using Gaussians placed at the positions of partials in the log-frequency domain and the synchronous evolution of partials belonging to the same source is modelled by Gaussian mixtures.

More recently, Peeling and Godsill, in their article *Multiple pitch estimation using non-homogeneous Poisson processes* [22], also proposed a likelihood function for multiple-pitch estimation where for a given time frame, the occurrence of peaks in the frequency domain is assumed to follow an inhomogeneous Poisson process. Also, Koretz and Tabrikian in *Maximum a posteriori probability multiple-pitch tracking using the harmonic model* [23], proposed an iterative method for multi-pitch estimation, which combines MAP and ML criteria. The predominant source is expressed using a harmonic model while the remaining harmonic signals are modelled as Gaussian interference sources [23].

1.2.3 Spectrogram factorisation-based multi-pitch detection

The majority of recent multi-pitch detection papers utilise and expand spectrogram factorisation techniques. Non-negative matrix factorisation (NMF) is a technique first introduced in their paper by Paris Smaragdis et al. [24] as a tool for music transcription. In its simplest form, the NMF model decomposes an input spectrogram $\mathbf{X} \in \mathbb{R}_+^{K \times N}$ with K frequency bins and N frames:

$$\mathbf{X} \approx \mathbf{W}\mathbf{H}$$

where $R \ll K, N$; $\mathbf{W} \in \mathbb{R}_+^{K \times R}$ contains the spectral bases for each of the R pitch components; and $\mathbf{H} \in \mathbb{R}_+^{R \times N}$ is the pitch activity matrix across time.

In his paper *Realtime multiple pitch observation using sparse non-negative constraints*, Cont Arshia applies NMF to AMT problem. Sparseness constraints were added into the NMF update rules, in order to find meaningful transcriptions using a minimum number of non-zero elements in \mathbf{H} . Emmanuel Vincent et al. in their article *Adaptive harmonic spectral decomposition for multiple pitch estimation* [25] incorporated harmonicity constraints in the NMF model, resulting in two algorithms: harmonic and inharmonic NMF. The inharmonic version of the algorithm is also able to support deviations from perfect harmonicity and standard equal temperament tuning. Also, Nancy Bertin et al. in their article [26] proposed a Bayesian framework for NMF, which considers each pitch as a model of Gaussian components in harmonic positions.

More recently, Ochiai et al. in his paper *Explicit beat structure modeling for non-negative matrix factorization-based multipitch analysis* [27] proposed an algorithm for multi-pitch detection and beat structure analysis. The NMF objective function is constrained using information from the rhythmic structure of the recording. It helped to improve transcription accuracy in highly repetitive recordings.

This thesis approaches AMT problem in similar fashion to spectrogram factorisation and feature-based methods. Detailed description of used methods is in Chapter 2.

1.3 Note Tracking

Typically AMT algorithms compute a time-pitch representation which needs to be further processed in order to detect note events with a discrete pitch value, a time of start and end of the note. This process is called *note tracking*. Most spectrogram factorisation-based methods estimate the binary piano-roll representation from the pitch activation matrix using simple thresholding (i.e. in *Explicit beat structure modeling for non-negative matrix factorization-based multipitch analysis* [28] by Graham Grindlay and in *Adaptive harmonic spectral decomposition for multiple pitch estimation* [25] by Emmanuel Vincent).

This approach will be used in the implementation of the thesis. Also, one simple and fast optimisation for note tracking is minimum duration pruning, which is applied after thresholding (idea comes from paper by Arnaud Dessein et al. *Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence* [29]). Primary idea is that output notes that have a duration smaller than a predefined threshold are removed from the final score. Similar method was also used by Juan Pablo Bello et al. in their paper *Automatic piano transcription using frequency and time-domain information* [30], where more complex rules for note tracking were used, addressing cases such as where a small gap exists between two note events. This method will also be applied as it is easy to implement.

For threshold based method, there are several issues that may appear for different instruments, primarily related to how notes are used and written for them in practice. For instance, for percussion instruments, note decay is exponential and physical duration of the note is irrelevant as it is not controlled (for most percussion instruments) by a musician. This way notes may appear short and require pauses (rests) after them, even though the rests would not be written in sheet music. Such problems may be solved with other rule based solutions specific to each instrument that requires them or more complex approaches.

Even though a simple threshold-based solution was used for the note tracking task, following paragraph discusses some more complex and more accurate solutions, though without detailed description of the approaches.

Hidden Markov models (HMMs) are frequently used at a stage of postprocessing of note tracking. In his work *A discriminative model for polyphonic piano transcription* [31], Graham Poliner proposes a note tracking method that utilizes pitch-wise HMMs, where each HMM has two states, indicating note activity and inactivity. HMM parameters (state transitions and priors) were learned directly from a ground-truth training set, while the observation probability is given by the posterigram output for a specific pitch.

In *Polyphonic music transcription using note event modeling* [32] by Matti P. Ryynanen and Anssi Klapuri, a feature-based multi-pitch detection system was combined with a musicological model for estimating musical key and note transition probabilities. Note events are described using 3-state HMMs, which model the envelope (attack, sustain, and noise/silence states) of each sound. In addition, context-dependent HMMs were employed in *Automatic transcription of recorded music* [16] for determining note events by combining the output of a multi-pitch detection system with a note-start detection system.

Finally, dynamic Bayesian networks (DBNs) were proposed by Shigeki Sagayama et al. in their paper [33] for note tracking. They used the pitch activation of a NMF-based multi-pitch detection algorithm as input. The DBN has a note layer in the lowest level, followed by a note combination layer. Model parameters were learned using MIDI files from F. Chopin piano pieces.

1.4 Tuning, time signature, key, and tempo estimation

There are several other subtasks of AMT systems that have to be resolved to be able to generate correct transcription in a form of sheet music. Also, such estimates, if properly incorporated to the system, may improve estimates of detected pitches and their durations, events. Tuning, time signature, key, and tempo estimation are such tasks.

1.4.1 Key and chord detection

Most Western music has a harmonic organisation around one key. The key is generally unchanged over whole, or at least sections of musical pieces. At one point in time, the harmony may be described by chord, which is a combinations of simultaneous or sequential notes which are perceived to belong together (and in general sound nice together, even though any combination of notes has its chord). Algorithms for key (and similarly for chord) detection use template matching or HMMs. For key detection, this thesis uses the simple approach defined in Section 2.9.

1.4.2 Tempo and time signature estimation

The *beats* are regularly spaced in time pulses. They are the primary unit that defines tempo and rhythm of most Western music. A number of beats per unit of time (commonly per minute in sheet music) defines a *tempo*. A number of beats per uniform repetitive sections in score (bars) defines a *time signature*. In order to interpret an audio recording in terms of such a structure (which is necessary in order to produce Western music notation), the first step is to determine the rate of the most salient pulse, which is the tempo.

Algorithms used for tempo induction include autocorrelation, Fourier transforms, and periodicity transform, which are applied to audio features such as a note-start detection function (as Fabien Gouyon and Simon Dixon describe in their article *A review of automatic rhythm description systems* [34]). The next step involves estimating the timing of the beats constituting the main pulse, a task known as beat tracking. Again, numerous approaches have been proposed, such as rule-based methods (as in *Computational models of beat induction: The rule-based approach* [35] by Peter Desain and Henkjan Honing), adaptive oscillators (as in *Resonance and the perception of musical meter* [36] by Edward W Large and John F Kolen), agent-based or multiple hypothesis trackers (as in *Automatic extraction of tempo and beat from expressive performances* [37] by Simon Dixon), and other.

Böck et. al. proposed a novel tempo estimation algorithm based on a recurrent neural network that learn an intermediate beat-level representation of the audio signal which is then fed to a bank of resonating comb filters to

estimate the dominant tempo [38]. This algorithm got the best score in ISMIR 2015 Audio Tempo Estimation task. The implementation by the authors is included in the opensource *Madmom* audio signal processing library⁵ which will be used in the implementation.

The final step for rhythmic analysis consists of estimating the time signature, which indicates how beats are grouped and subdivided at respectively higher and lower metrical levels, and assigning each note-start and offset time to a position in this structure [39].

⁵<https://pypi.org/project/madmom/>

Analysis and design

This chapter defines architecture of the chosen solution. It provides details of used approaches for music sources separation and models used in it, pitch extraction and signal analysis, event detection, etc.

2.1 Architecture

The implementation of the system is separated into logical parts responsible for sound data streaming, music source separation, pitch and events detection, transcription and score generation. Following diagram shows architecture of the solution. Arrows represent data flow. Dotted arrows represent flow that is optional. If given parameters (like tuning, tempo, time signature and key) are specified by user, they are not being estimated. Each rectangular block represents logical module in implementation.

Detailed description of each component is in the dedicated section following the diagram in Fig. 2.1.

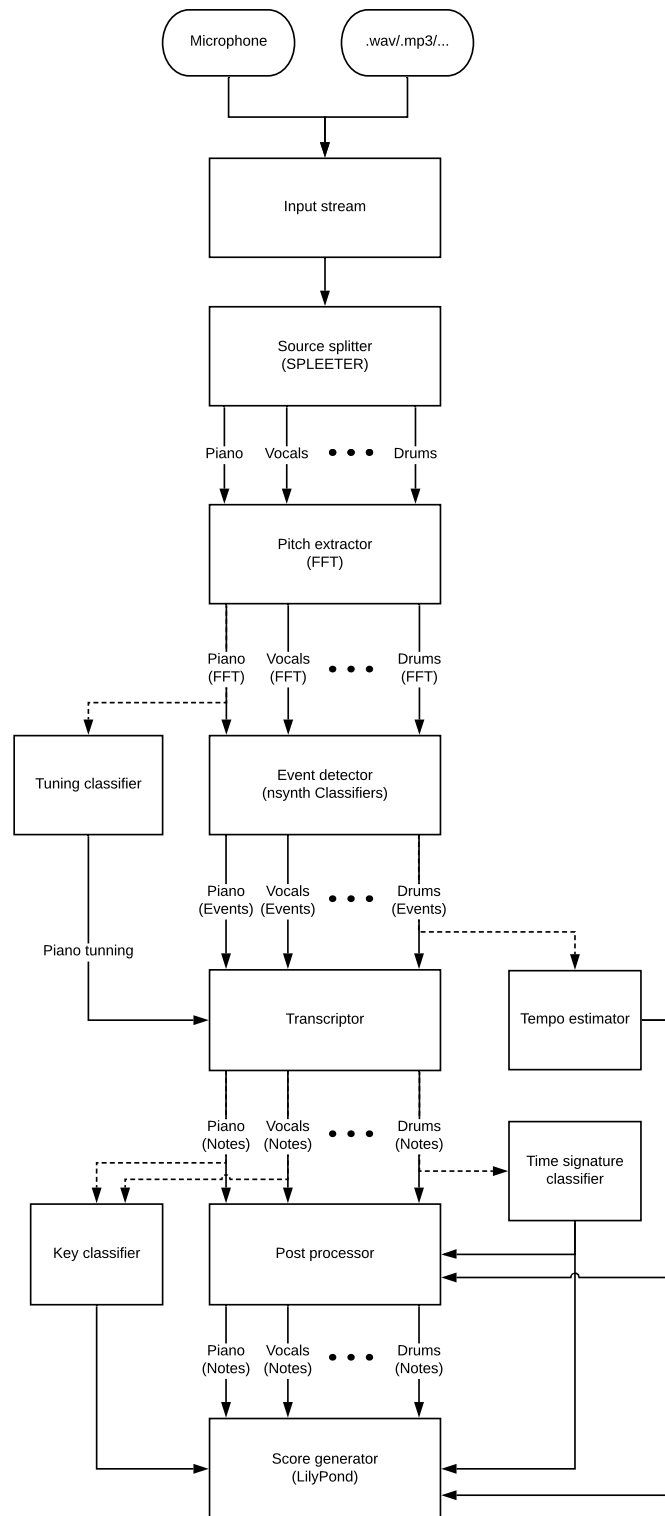


Figure 2.1: Architecture of the implementation.

2.2 Audio streaming

This implementation directly works only with WAVE (.wav/.wave). Any other format is converted to WAVE first, then processed.

2.2.1 WAVE format

WAVE is an audio file format standard, developed by Microsoft and IBM, for storing an audio bitstream on PCs. What's important for this thesis and implementation is that it stores data in chunks in Linear pulse-code modulation (LPCM) format. This format allows to perform Discrete Fourier transform (DFT) used in pitch extraction.

2.2.2 Sampling rate

LPCM mentioned above stores sampled amplitude of recorded audio at specific sampling rate (frequency, measured in Hz).

The most common sampling rate is 44.1 kHz, or 44100 samples per second. This is the standard for most consumer audio, used for formats like CDs [40].

The sampling rate determines the range of frequencies captured in digital audio. The lowest frequency a person can hear is 20 Hz. The highest frequency humans can hear are in the range of 20.000 Hz, but only young people can hear such high tones [41]. According to *Nyquist Theorem*, a signal which has a Fourier transform having only frequencies upto a certain maximum f_m , we can obtain the analog signal $f(t)$ from the sampled signal $f'(t)$ by passing the sampled signal $f'(t)$ through a low pass filter provided that the sampling frequency f_s is more than twice the maximum frequency f_m present in the signal i.e. , $f_s > 2f_m$ [42]. Hence, having 44100 Hz sampling rate, we can reproduce and analyse frequencies up to 22050 Hz (assuming an ideal low pass filter). Otherwise, if recorder has a sampling rate lower than $2\times$ the highest frequency (which was not cut off by low pass filter) it causes the effect called *aliasing*, which introduces unexpected sounds in the recording that were not present in the original sound. If the sampling frequency is too low the frequency spectrum overlaps, and becomes corrupted [42].

The implementation is able to process input sound with any sampling rate, though lower sampling limits processed frequencies range to lower pitches.

2.3 Music source separation

First step of the sound processing is separation of the sound into source instruments (i.e. voice, guitar, piano, etc.)

As was mentioned in the previous chapter, this implementation uses *Spleeter* for separation of source instruments. *Spleeter* is a fast and state-of-the art music source separation tool with pre-trained models [2]. Its implementation contains three pre-trained models:

- vocals/accompaniment separation,
- 4 stems separation as in SiSeC [5] (vocals, bass, drums and other),
- 5 stems separation with an extra piano stem (vocals, bass, drums, piano and other). It is, to the author's knowledge, the first released model to perform such a separation.

Estimations for all the models is performed in a frequency domain of the sound. Meaning that sound data from time domain is converted to frequency domain using FFT, passed to the models described in section 1.1.2 about U-net architecture. Output of the model is separated tracks for each instrument and voice. To get sound of each instrument and voice in time domain (as it would be represented in WAVE), we would need to pass it through Inverse Discrete Fourier transform (IDFT). But it is not necessary, as all the subsequent processing will be performed on the sound in frequency domain.

More about FFT is in the following section 2.4 about pitch extraction.

2.4 Pitch extraction

Section 1.2 analyses many approaches to pitch (and specifically to multi-pitch) detection. The one that is presented in this theses utilizes a combination of ideas defined in works of Matti P Ryyanen et al. [32], Arnaud Dessein et al. [29] and Paris Smaragdis et al. [24]. Solution is joint, thus estimates played notes at a given moment all at once (opposed to iterative approaches). It attempts to detect frequencies similarly to matrix factorization techniques through analysis of sound spectrogram. But instead of matrix factorization (NMF), this work attempts to detect notes' events, specifically their envelopes (more on the sound envelope in Section 2.4.1), using ML models. Specification of the used data and training of the models is in Section 2.5.1.

2.4.1 Sound envelope

Sound envelope is a variation of the sound volume in time [43]. Sound envelope consists of 4 stages: attack, decay, sustain, and release (ADSR):

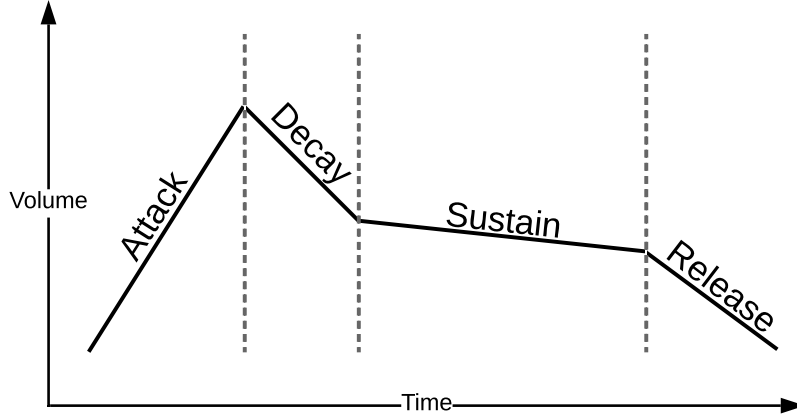


Figure 2.2: Sound envelope.

Fig. 2.2 shows a theoretical simple ADSR model of sound envelope. But different instruments produce different envelopes depending on a nature of sound extraction:

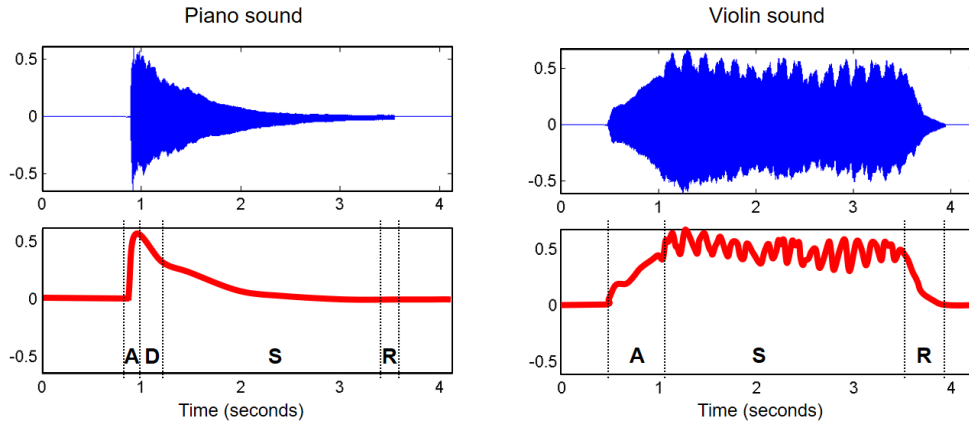


Figure 2.3: Sound envelopes of piano and violin [44].

As seen in Fig. 2.3, piano and any other instrument that produces sound by hitting, tapping or pinching of a string (like guitar, harp, bandura, balalaika, etc.), will produce similar envelope with defined attack (the moment of piano

key pressing; pinching or hitting a string on guitar, etc.), decay and sustain (when piano key remains pressed or piano sustain pedal is used, etc.) and release (when piano key and sustain pedal are released, guitar strings are muted, etc.).

2.5 Event detection

Note and subsequently its pitch and start are estimated by detecting its envelope. Having a sample of sound (change of volume of each pitch as determined from FFT) of duration k seconds specified by implementation, predictive model attempts to estimate whether note has been played at a given point in time by detecting its envelope that should look similar for each note of the given instrument. That means that there will be a model for each predefined instrument trained on its samples (more in Section 2.5.1).

2.5.1 Data and model training

Dataset for training of the above-mentioned models was generated from the *NSynth dataset* [45]. *NSynth* is an audio dataset containing 305,979 musical notes, each with a unique pitch, timbre, and envelope. For 1,006 instruments from commercial sample libraries, there are generated four second, monophonic 16kHz audio snippets, referred to as notes, by ranging over every pitch of a standard MIDI piano (21-108) as well as five different velocities (25, 50, 75, 100, 127) [45].

NSynth contains samples for 11 different instruments: bass, brass, flute, guitar, keyboard, mallet, organ, reed, string, synth_lead, vocal. They are all stored in WAVE format and have needed metadata in JSON format alongside with them. Metadata for each sample includes instrument, note, pitch and velocity in Musical Instrument Digital Interface (MIDI) format (in the range [0, 127]), and sampling rate.

Spleeter, used for source separation, is able to separate sound only into 5 source instruments. Hence only those samples from NSynth will be used to generate models.

The preprocessing of the training dataset is completely the same as preprocessing of the sound samples during transcription. The whole data flow is shown in Fig. 2.4.

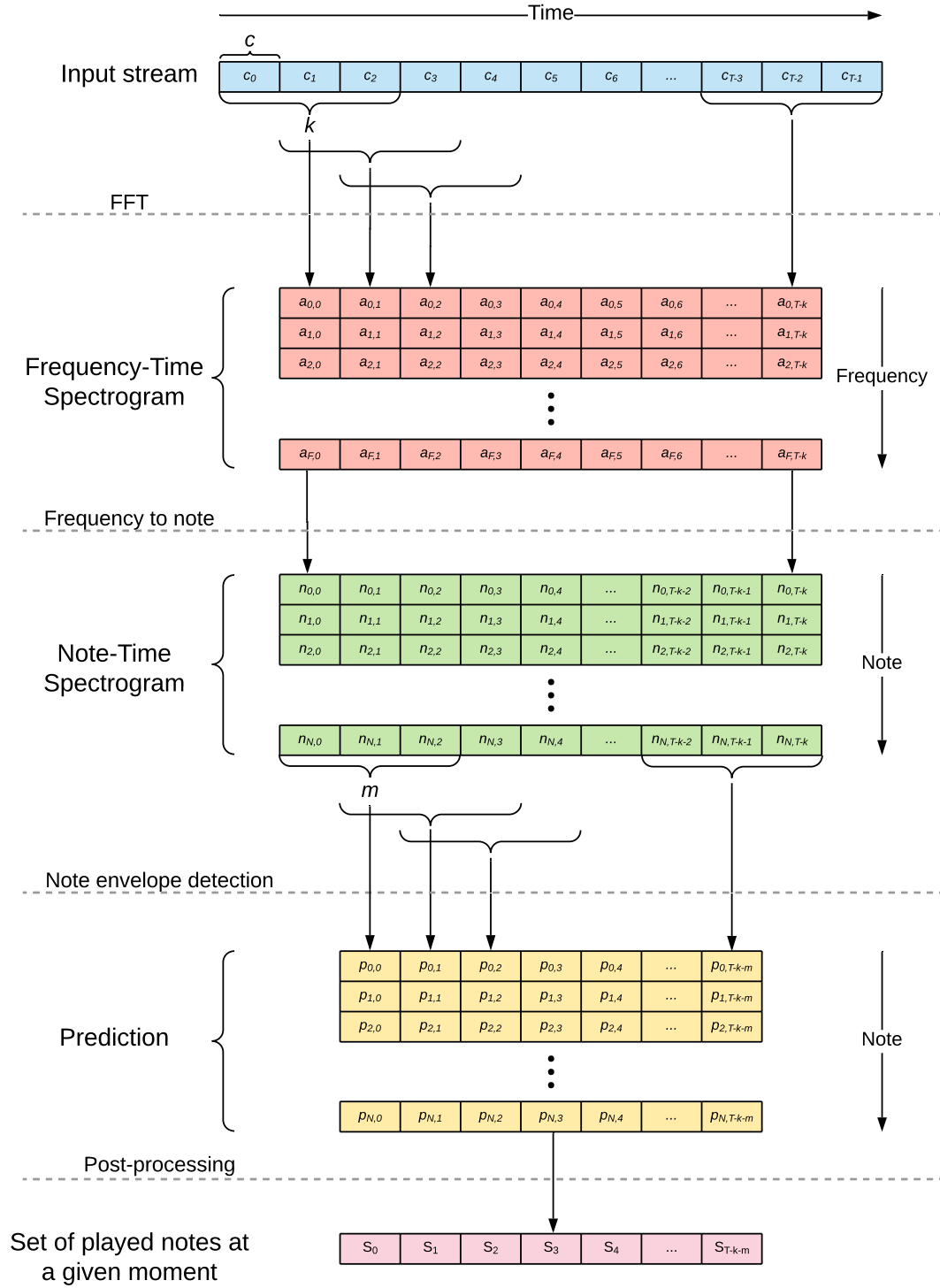


Figure 2.4: Data flow for pitch detection.

As shown in Fig. 2.4, input stream (blue) is a stream of data read from input file or microphone (or any other input). It is read by chunks of size c determined by implementation. Each window of k chunks is passed through FFT to transform data to a frequency domain. Taking several chunks of data to pass it through FFT increases its accuracy, peaks of played pitches become more prominent and output becomes more robust to noise and phase shifts. Overlapping of k -sized windows allows producing more data-points per second and subsequently features for predictive model. Having T chunks and window of size k , produced spectrogram is of size $F, T - k$ where F is a number of detected frequencies.

After transforming input to time-frequency spectrogram (red), frequencies are translated to musical notes (green). Assuming equal temperament tuning and A with 440 Hz frequency (actual tuning will be estimated later in the analysis), frequencies are converted to the closest note. Frequencies converted to the same note are filtered to leave only the highest volume value.

The output of previous step is passed to the model of a given instrument by window of size m . So m is a number of input features of the model. The model attempts to classify whether given window contains an envelope of a played sound that starts from a given point in time. So for $\forall i \in [0, N], j \in [0, T - k - m]$, $p_{i,j}$ (pink) shows prediction of the model for note i at a time j .

Training data is generated from NSynth dataset in a similar fashion. Positive labels are set for the pitch being played in a sample, negative for all the others. Also negative examples are generated from the same sample for played pitch but with a time shift, starting the example from or ending it somewhere in the middle of the actual sound envelope.

2.6 Tuning classification

Tuning of the instrument is not a part of score transcribed into sheet music and most of the Western music follows the same tuning system. Specifically, equal temperament system with A tuned to 440 Hz. But tuning estimation is an essential part for correct reproduction of the sound.

There are two primary parts to tuning estimation: detection of frequency of base note, commonly A :



Figure 2.5: Base A commonly tuned to 440 Hz.

and tuning system, like equal temperament, pythagorean, meantone, etc. While tuning system in the Western music very rarely diverges from equal

temperament, frequency of a base note can often be chosen to be different from 440 Hz. That also often might happen for instruments that are often being retuned like guitars as tuning “by ear” by person that does not have a perfect pitch⁶ is defined by tuning of the string relative to which all other strings are tuned.

Having found one played frequency F in sound (or several frequencies for better precision), it is matched to the closest note N in 440-Hz-A-tuning, then real frequency of A note $f(A)$ is calculated as:

$$f(A) = F * 2^{\frac{t(N)-t(A)}{12}}$$

where $t(N)$ is an index number of the note(semitone) N (for example in MIDI representation). Now, all the other notes can be calculated in the same way.

2.7 Tempo estimation

As was mentioned in Section 1.4.2, *Madmom* library will be used for the task of tempo estimation. Authors use a recurrent neural network to learn an intermediate beat-level representation of the audio signal. The output of the neural network is a beat activation function, which represents the probability of a frame being a beat position. And instead of processing the beat activation function to extract the positions of the beats, authors use it directly as a one-dimensional input to the bank of resonating comb filters. Comb filtering can be defined as “the frequency response caused by combining a sound with its delayed duplicate. The frequency response displays a series of peaks and dips caused by phase interference. The peaks and dips look like the teeth in a comb, with very narrow, deep notches where signals are attenuated.” [47]. Using comb filters with different lags (delays) implementation of *madmom* detects at which lag the beat of the sound resonates the most. Given lag would then define a tempo.

The range of possible tempo values (beats per minute (BPM)) t is limited to $1 \leq t \leq 128$ and to whole numbers only. This is decided so that the range can include loops that last from only 1 beat to 128 beats, which would correspond to a maximum of 32 bars in $\frac{4}{4}$ meter.

2.8 Time signature estimation

Problem of time signature or *meter* estimation is similar to tempo detection in a sense that the basic idea of it is finding its repetitiveness, recurrence -

⁶*Perfect pitch* or *absolute pitch* is the ability to identify a note by hearing it. The ability is considered remarkably rare, estimated to be less than one in 10,000 individuals [46].

beat for tempo and content of a sheet music bar for time signature. That is why solutions for these problems often overlap.

Another approach for tempo as well as for time signature estimation is autocorrelation modeling. Autocorrelation modeling is used to determine the length of the bar - number of beats per each meter. Technically, time signature definition can contain any numbers for number of beats per measure (top number) and the note value that receives one beat (bottom number). But most of music peaces of western music use powers of 2 as a note value (otherwise it is called irrational measure) and rarely higher than 8 (♩). As for number of beats per bar, a 1 or values higher than 12 are considered to be the unusual time signatures. So the implementation limits estimation to these ranges.

Knowing the tempo - a number of fourth notes (♩) per minute, taking an average volume of the notes (0 if no notes are there) in all position in time of sixteenth notes (♩) produces the time series on which implementation models autocorrelation. Assuming that rhythmical structure of the bar and position of strong and weak⁷ beats is continues through the whole peace or its significant part, the lag of modeled autocorrelation will determine the number of sixteenth notes per bar.

Autoregressive integrated moving average (ARIMA) model is used to determine the lag. ARIMA is a class of models that “explains” a given time series based on its own past values, that is, its own lags (AR part) and the lagged forecast errors (MA part), so that equation can be used to forecast future values. Specifically its simpler version AR will be used. AR is defined as follows:

$$Y_t = \alpha + \beta_1 * Y_{t-1} + \beta_2 * Y_{t-2} + \dots + \beta_n * Y_{t-n} + \epsilon_1$$

where Y_t is value measured in time t , α is the intercept term, β_k is coefficient of the first lag, and ϵ is a noise. All of β s and α are estimated by the model. The higher the absolute value β_k - the higher the correlation between the signal and its copy delayed by lag k . Obviously the highest correlation of a signal is with its 0 lag. But as was explained above, the choice is limited to range $2 \leq k \leq 12$ with 8 as the shortest note value. So the coefficients β are estimated from 8th up to 32nd lag to determine time signatures from $\frac{2}{4}$ (which in time is equivalent to $\frac{8}{16}$) up to $\frac{12}{8}$ (which in time is equivalent to $\frac{24}{16}$) and up to $\frac{8}{4}$ or $\frac{4}{2}$ (which in time are equivalent to $\frac{32}{16}$).

Technically any score for which appropriate k was found, can be written within $\frac{k}{16}$ measure. But it is better to identify the best logical value for the number of beats per measure for simplification of reading of the score and convert the time signature to either $\frac{k/2}{8}$, $\frac{k/4}{4}$, or $\frac{k/8}{2}$.

⁷Commonly, some notes per bar are *strong*(louder) and some are *weak*(quieter). This determines accents in measure. For example in $\frac{4}{4}$ time, first beat is often the loudest(strong), third is also strong, but not as strong as the first, and second and fourth are weak.

k	16 (♩)	8 (♩)	4 (♩)	2 (♩)
8	$\frac{8}{16}$	$\frac{4}{8}$	$\frac{2}{4}$	$\frac{1}{2}$
10	$\frac{10}{16}$	$\frac{5}{8}$		
12	$\frac{12}{16}$	$\frac{6}{8}$	$\frac{3}{4}$	
14	$\frac{14}{16}$	$\frac{7}{8}$		
16	$\frac{16}{16}$	$\frac{8}{8}$	$\frac{4}{4}$	$\frac{2}{2}$
18	$\frac{18}{16}$	$\frac{9}{8}$		
20	$\frac{20}{16}$	$\frac{10}{8}$	$\frac{5}{4}$	
22	$\frac{22}{16}$	$\frac{11}{8}$		
24	$\frac{24}{16}$	$\frac{12}{8}$	$\frac{6}{4}$	$\frac{3}{2}$
26	$\frac{26}{16}$	$\frac{13}{8}$		
28	$\frac{28}{16}$	$\frac{14}{8}$	$\frac{7}{4}$	
30	$\frac{30}{16}$	$\frac{15}{8}$		
32	$\frac{32}{16}$	$\frac{16}{8}$	$\frac{8}{4}$	$\frac{4}{2}$

Table 2.1: Time signature selection table.

Having found a value of k , conversion is performed according to a Table 2.1 as indicated by green cells for even values of k . Odd values of k are not expected as they are very unusual in measures of $\frac{k}{16}$. If they are odd, the time signature is left as is. It is important to note that measures indicated by green cell are more common than their counterparts within a row but time signatures like $\frac{6}{8}$ and $\frac{2}{2}$ are also widely used in music. But it is hard to objectively identify which of the measures $\frac{6}{8}$ or $\frac{3}{4}$, $\frac{2}{2}$ or $\frac{4}{4}$ should be used.

2.9 Key classification

Key signature is a part of sheet music notation that simplifies it by avoiding redundant repetitive accidentals (sharps and flats) and by defining the set of used notes which in its turn defines the set of used chords, their progressions and harmonic functions in a piece of music. Detailed description of key signature is in Appendix B.3.6.

As was mentioned in Section 1.4.1, there are several approaches to key classification including template matching or HMMs. The best paid solution for key detection is *Mixed In Key* [1] having 95% accuracy on their test dataset. The best free solution is *KeyFinder* [48] with accuracy of 77%. But it is implemented in C++ so is hard to incorporate into Python implementation

used in this work.

A simple heuristical solution was used in the framework of this thesis. For each note that has a sharp counterpart (*C*, *D*, *F*, *G*, and *A*), if its semitone-higher (sharper) note appears more often than its natural note in any octave, its sharp symbol is included into a key signature. For example:



Figure 2.6: Notes in a key of *C major*.

will be translated into



Figure 2.7: Notes in a key of *G major* (note a # on the *F* line).

The notes above are one *F* (natural) and two of *F#* from three different octaves. As there are more *F#* notes, output transcription will be in a key of *G major*, that consists of notes *G*, *A*, *B*, *C*, *D*, *E*, and *F#*.

It is important to note that *E minor* key has the same set of notes, but to determine whether key is *G major* or *E minor* is a much more complex task and requires an analysis of chord progressions and their harmonic functions within the context of a given score.

2.10 Post processing

Duration of the note is determined by its start (start of the sample passed to the model) and its end (moment, when note's volume lowers under the predefined threshold). As was discussed in Chapter 1, implementation also utilizes several simple postprocessing ideas:

- if the duration of the note is too short, it is omitted,
- if the note played at the same time with another note but with the significant difference in volume, the quieter note is omitted.

Another goal of post processing module is the determination of the rests positions and their lengths. Rests fill in the gaps in a bar where no note is played. They are required to position notes correctly on the staff and

subsequently in time for the musician. As for notes, it is needed to determine time of the rests' start and their length (whole rest, half rest, quarter rest, etc.). The process is fairly straightforward having the notes and their positions detected:

```

for each bar do
  while not the end of the bar do
    if any note starts at the current position then
      | go to the end of the note;
    else
      | mark start of the rest;
      | go to the next note or an end of the bar whatever comes
      | first and mark it as an end of the rest;
      | determine the length of the rest from its start and end;
    end
  end
end

```

Figure 2.8: Rests detection algorithm.

2.11 Score generation

Finally, having estimated time signature, key, positions and pitches of notes, rests, the output transcription is generated.

LilyPond [49] is used for score generation. Lilypond is a computer program and file format for music engraving. Notes in Lilypond are represented in pitch-duration format: pitch is specified with *Helmholtz pitch notation*⁸, and duration is specified with a *numeral based system*⁹.

The output of this module is a Lilypond file (in .ly format) and subsequently score in Portable Document Format (PDF).

⁸*Helmholtz pitch notation* is a system for naming musical notes. For example, the note C is shown in different octaves by using upper-case letters for low notes, and lower-case letters for high notes, and adding sub-primes and primes in the following sequence: $C_{//}$ C_1 C c c' c'' c''' , where c' is the *middle C*.

⁹Duration of a note is specified with numbers 1 (♩), 2 (♪), 4 (♫), 8 (♮), etc. For a quarter note (♫) number can be omitted.

Implementation

This chapter provides details of the implementation, used tools, training and testing of the models.

3.1 Used tools

Python was used as a primary programming language for implementation of the project. Input streams are processed by *pyaudio* library. Data is stored in a *numpy* array in 16 bit integers. *Numpy* and *Pandas* were used for datasets processing. *Scipy*'s implementation of deep neural networks (DNNs) (and other models that were tested) was used for models of event detection module. Models were trained in *jupyter notebook* included in the implementation sources. Models are serialized into .pickle format using Python's *joblib* module. The *statsmodels* library is used to generate ARIMA model used in time signature estimation

As was discussed in Chapter 2, implementations of *Spleeter* and *Madmom* were used for source splitting and tempo detection modules respectively. Output score in lilypond format is generated using *abjad* that has Pythonic object-oriented interface for sheet music engraving which uses the lilypond compiler.

3.2 Project structure

Project follows common approaches for Python project structures and implements a Python module as well as command line interface (CLI) for music transcription. It is well parametrized such that user can define a set and tuning of the instruments, tempo, time signature and key. Otherwise these parameters are being estimated by dedicated modules according to design described in a Chapter 2.

3. IMPLEMENTATION

The main endpoint that is used by CLI and may be used by developers is *mimt.music_transcription* (mimt stands for multi-instrument music transcription). It is responsible for the whole data pipeline that starts at input stream reading.

3.2.1 Input stream

Input stream can have any source: file, microphone, or any other source which implements *audio_reading.AbstractStream* interface. For microphone and file streaming there are already implemented classes *MicrophoneStream* and *FileStream* respectively. Implementations are parametrized by size of read chunks, but it is important to note that divergence from default value will affect the representation of the data which in its turn would require retraining of the models used in event detection.

3.2.2 Music source separation and tempo estimation

Modules of the implementation utilize existing solutions for problems of source separation (*Spleeter*) and tempo estimation (*Madmom*). There are a dedicated modules for both of the problems. The modules are just calling needed functions from implementations of *Spleeter* and *Madmom*. Hence, modules are simple and were included into a project as a dedicated Python modules only to be consistent within a structure and interfaces of the implementation.

By default, this implementation uses five stems separation. Otherwise it can be defined as a parameter of the analysis. If any of the output sheet music scores does not have any notes, it is ignored and does not have output sheet music.

3.2.3 Pitch detection

Following the source separation module, pitch extraction transforms signals of each instrument from time domain to a frequency domain using FFT. Numpy's *fft* submodule is used to perform the transformation. Amplitudes of frequencies generated by FFT then converted to decibels with the following formula:

$$volume = 10 * \log_{10} amplitude$$

PitchExtractor stores the frame rate of the input stream to correctly convert frequencies produced by FFT to real frequencies of a sound (Hz):

$$frequency = fft_frequency * frame_rate$$

Frequencies are then converted to the closest notes. Assuming equal temperament tuning system and note A tuned to 440 Hz, each frequency is converted to a note in a format used in *abjad* modules, specifically *{note_name}{octave}*, e.g. *A2* is 440 Hz A note, *C2* is the *middle C*, *Cs3* is a *C♯* - one octave and

a semitone higher of the *middle C*, etc. First, the semitone number above or below *440-Hz-A* is calculated:

$$n = \lfloor \log_{12} \frac{frequency}{440} \rfloor$$

where $n = 0$ for the *440-Hz-A*, $n = 1$ for *A♯*, $n = -9$ for the *middle C*, etc.

The *note_name* and *octave* are then calculated simply:

$$note_name = NOTES[n \bmod 12]$$

where

$$NOTES = [A, As, B, C, Cs, D, Ds, E, F, Fs, G, Gs]$$


and

$$octave = 2 + \lfloor \frac{n}{12} \rfloor$$

where 2 is an octave of the *440-Hz-A*.

Only the maximum volume value is selected from the frequencies that are converted to the same note.

3.2.4 Event detection

EventDetector from *event_detection* module goes through the spectrogram (time-volume representation of the notes) of the sound with overlapping window as shown in Fig. 2.4 (green). The size of the window is determined by 3 parameters: shortest note value (default is 16()), sampling rate of input stream, and overlapping rate which defines how many data points (volume in time) are shared between two consequent windows. Having these 3 parameters, the size of the window is calculated as

$$window_size = sampling_rate / (shortest_note_value / overlapping_rate)$$

and step between windows as

$$window_step = \lfloor sampling_rate / shortest_note_value \rfloor$$

The window of spectrogram is passed to the pretrained model for the given instrument to detect whether this window contains the envelope(s) for the note(s) in this point in time. If it does, this window constitutes the start of the note. As was mention in Section 2.5, the end of the detected note is a point in time when its volume drops under specified threshold *note_volume* - 10.

Testing

4.1 Source separation

Commonly used source separation quality evaluation metrics are presented in a paper by Emmanuel Vincent et al. [50]. For a separating performance measures are computed for each estimated source \hat{s}_j by comparing it to a given true source s_j . The computation of the criteria involves two successive steps. In a first step, they decompose \hat{s}_j as:

$$\hat{s}_j = s_{target} + e_{interf} + e_{noise} + e_{artif}$$

where $s_{target} = f(s_j)$ is a version of s_j modified by an allowed distortion $f \in F$, and where e_{interf} , e_{noise} and e_{artif} are the interferences, noise, and artifacts error terms respectively. These four terms represent the parts of \hat{s}_j that come from the real source s_j , from unwanted sources $(s_i)_{i \neq j}$, from noise, and from other causes. The performance of the model then is evaluated by 4 metrics:

- Source to Distortion Ratio (SDR) = $10 \log_{10} \frac{\|s_{target}\|^2}{\|e_{interf} + e_{noise} + e_{artif}\|^2}$
- Source to Interferences Ratio (SIR) = $10 \log_{10} \frac{\|s_{target}\|^2}{\|e_{interf}\|^2}$
- Sources to Noise Ratio (SNR) = $10 \log_{10} \frac{\|s_{target} + e_{interf}\|^2}{\|e_{noise}\|^2}$
- Sources to Artifacts Ratio (SAR) = $10 \log_{10} \frac{\|s_{target} + e_{interf} + e_{noise}\|^2}{\|e_{artif}\|^2}$

The Spleeter's performance measured on the standard musdb18 dataset [6] comparing to *Open-Unmix* [51] implementation is shown in Table 4.1.

4. TESTING

	SDR	SIR	SAR	ISR	SDR	SIR	SAR	ISR
	vocals				bass			
Spleeter	6.88	15.86	6.99	12.01	5.51	10.30	5.96	9.61
Open-Unmix	6.32	13.33	6.52	11.93	5.23	10.93	6.34	9.23
	drums				other			
Spleeter	6.71	13.67	6.54	10.69	4.55	8.16	4.88	9.87
Open-Unmix	5.73	11.12	6.02	10.51	4.02	6.59	4.74	9.31

Table 4.1: Spleeter and Open-Unmix performances.

4.2 Sound envelope detection

Sound envelope detection models were trained and tested on the *nsynth* dataset [45]. For the purposes of the model selection, the dataset generated from the *nsynth* data was split into four cross validation folds. Performance results averaged among the folds are shown in Table 4.2.

model	time (s)		test (%)			train (%)		
	fit	score	accuracy	precision	recall	accuracy	precision	recall
MLP	6.22	0.05	65.0	70.8	57.3	67.3	72.4	60.6
RandomForest	3.72	0.48	64.6	66.6	61.7	1.00	1.00	1.00
XGB	2.45	0.08	64.3	67.0	60.3	67.1	70.1	63.2
AdaBoost	1.72	0.32	63.3	65.5	60.0	65.7	68.4	62.0
DecisionTree	0.25	0.03	59.4	59.4	59.4	59.9	59.8	60.0
Logistic	0.12	0.02	58.0	57.6	59.0	1.00	1.00	1.00
GaussianNB	0.01	0.02	57.0	59.6	52.6	57.9	60.3	53.4
KNeighbors	0.01	0.93	51.2	52.2	48.2	60.3	61.9	57.3

Table 4.2: Envelope detection models’ performance.

As seen from Table 4.2, the best performing model is *multilayer perceptron classifier*. It is a neural network with two hidden layers with 64 and 32 neurons respectively. It uses L2 regularization with $\alpha = 0.001$ and adam optimizer for learning.

The accuracy of 65% is not a competitive performance, so this part requires some tuning of hyperparameters, network topology, or a completely different approach.

4.2.1 Other modules

The *key* estimation module does not need a testing dataset to evaluate its performance. It is simple rule based solution. Its accuracy is defined by ratio of musical pieces (say k) that follow its key - have more sharp notes for notes that have sharp symbol in the key signature and the same for flat and natural notes; and ratio of usage of major keys (as a major key is assumed by default in the implementation) among scores (say p). Having those two ratios the accuracy of this solution is $k * p$. Although the choice of the major or corresponding to it minor key does not affect the key signature, so accuracy for key signature estimation in output sheet music is just k .

Madmom does not provide evaluation of performance of *tempo* estimation functionality. So it has been tested. The Free Music Archive (FMA) dataset [52] was used for evaluation of *madmom*'s tempo prediction. Even though tempo estimation seems like a regression problem, small inaccuracies of the predictions are equally as bad as big ones as they deteriorate the whole subsequent analysis. So the metric for the problem would be the accuracy that shows a ratio of correctly estimated tempos to the number of analysed songs. On the small version FMA dataset with 8000 30s songs, the accuracy of *madmom*'s tempo estimation was 81.3%.

Conclusion

This thesis introduces a reader to the problem of automated transcription of a multi-instrument sound recording to sheet music. It discusses state-of-the-art solutions to the tasks of a source instruments separation, pitch detection, key, and time signatures estimation, etc.

The goal has been reached at a sufficient level. The analysis and design of the implementation allow for future expansion and improvement of the quality of the processing pipeline.

Possible improvements

Implementation does not take into consideration the *dynamics* of the notes. Dynamic of a sound describes its amplitude or loudness (*pp*, *p*, *mf*, *f*, *ff*, etc.), its emotional intensity and change through time (crescendo, decrescendo). It is often an inalienable part of the description of the generated sound, hence a part of the output sheet music. The solution to this task could be simple rule based or more complex that utilize ML or statistical analysis.

Another optimization for pitch detection could be usage of *Multiresolution FFT*. This method is widely-used in a field of Music Information Retrieval (MIR) tasks. The frequency components of a DFT are equally spaced and have a constant resolution. However, in polyphonic music a higher frequency resolution is needed in the low and mid frequencies where there is a higher density of harmonics. On the other hand, frequency modulation gets stronger as the number of harmonic is increased, requiring shorter windows for improved time resolution [53]. Thus, a multi resolution spectral representation is highly desired for the analysis of music signals and can be a great improvement for the pitch detection module of this implementation.

Bibliography

- [1] Vorobyev, Y. 2005, [Cited 2019-12-23]. Available from: <https://mixedinkey.com/>
- [2] Hennequin, R.; Khlif, A.; et al. Spleeter: A Fast And State-of-the Art Music Source Separation Tool With Pre-trained Models. Late-Breaking/Demo ISMIR 2019, November 2019, Deezer Research.
- [3] Jansson, A.; Humphrey, E.; et al. Singing voice separation with deep U-Net convolutional networks. October 2017. Available from: <https://openaccess.city.ac.uk/id/eprint/19289/>
- [4] Pretet, L.; Hennequin, R.; et al. Singing Voice Separation: A Study on Training Data. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, doi:10.1109/icassp.2019.8683555. Available from: <http://dx.doi.org/10.1109/ICASSP.2019.8683555>
- [5] Stöter, F.-R.; Liutkus, A.; et al. The 2018 Signal Separation Evaluation Campaign. 2018, 1804.06267.
- [6] Rafii, Z.; Liutkus, A.; et al. The MUSDB18 corpus for music separation. Dec. 2017, doi:10.5281/zenodo.1117372. Available from: <https://doi.org/10.5281/zenodo.1117372>
- [7] Liutkus, A.; Stöter, F.-R.; et al. The 2016 Signal Separation Evaluation Campaign. In *Latent Variable Analysis and Signal Separation - 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings*, edited by P. Tichavský; M. Babaie-Zadeh; O. J. Michel; N. Thirion-Moreau, Cham: Springer International Publishing, 2017, pp. 323–332.
- [8] Dumoulin, V.; Visin, F. A guide to convolution arithmetic for deep learning. 2016, 1603.07285.

- [9] Chunghsin Yeh; Robel, A.; et al. Multiple fundamental frequency estimation of polyphonic music signals. In *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 3, March 2005, ISSN 2379-190X, pp. iii/225–iii/228 Vol. 3, doi:10.1109/ICASSP.2005.1415687.
- [10] A. Degani, P. M., R. Leonardi. A pitch salience function derived from harmonic frequency deviations for polyphonic music analysis. Sept. 2014. Available from: http://www.cmap.polytechnique.fr/~bacry/MVA/getpapers.php?file=multipitch_2.pdf&type=pdf
- [11] Jones, C. Music Information Retrieval Evaluation eXchange(MIREX). 2010, [Cited 2020-01-05]. Available from: https://www.music-ir.org/mirex/wiki/MIREX_HOME
- [12] Kumar, S.; Stephan, K. E.; et al. Hierarchical processing of auditory objects in humans. *PLoS computational biology*, volume 3, no. 6, 2007: p. e100.
- [13] Dressler, K. Multiple fundamental frequency extraction for MIREX. *Eighth Music Information Retrieval Evaluation eXchange (MIREX)*, 2012.
- [14] Pertusa, A.; Inesta, J. M. Multiple fundamental frequency estimation using Gaussian smoothness. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2008, pp. 105–108.
- [15] Reis, G.; Fonseca, N.; et al. Hybrid genetic algorithm based on gene fragment competition for polyphonic music transcription. In *Workshops on Applications of Evolutionary Computation*, Springer, 2008, pp. 305–314.
- [16] Grosche, P.; Schuller, B.; et al. Automatic transcription of recorded music. *Acta Acustica united with Acustica*, volume 98, no. 2, 2012: pp. 199–215.
- [17] Nam, J.; Ngiam, J.; et al. A Classification-Based Polyphonic Piano Transcription Approach Using Learned Feature Representations. In *Ismir*, 2011, pp. 175–180.
- [18] Humphrey, E. J.; Bello, J. P.; et al. Feature learning and deep architectures: new directions for music informatics. *Journal of Intelligent Information Systems*, volume 41, no. 3, 2013: pp. 461–481.
- [19] Emiya, V.; Badeau, R.; et al. Multipitch Estimation of Piano Sounds Using a New Probabilistic Spectral Smoothness Principle. *IEEE Transactions on Audio, Speech, and Language Processing*, volume 18, no. 6, Aug 2010: pp. 1643–1654, ISSN 1558-7924, doi:10.1109/TASL.2009.2038819.

-
- [20] Goto, M. A real-time music-scene-description system: predominant-F0 estimation for detecting melody and bass lines in real-world audio signals. *Speech Communication*, volume 43, no. 4, 2004, ISSN 0167-6393, doi:<https://doi.org/10.1016/j.specom.2004.07.001>, special Issue on the Recognition and Organization of Real-World Sound. Available from: <http://www.sciencedirect.com/science/article/pii/S0167639304000640>
- [21] Kameoka, H.; Nishimoto, T.; et al. A Multipitch Analyzer Based on Harmonic Temporal Structured Clustering. *IEEE Transactions on Audio, Speech, and Language Processing*, volume 15, no. 3, March 2007: pp. 982–994, ISSN 1558-7924, doi:10.1109/TASL.2006.885248.
- [22] Peeling, P. H.; Godsill, S. J. Multiple pitch estimation using non-homogeneous Poisson processes. *IEEE Journal of Selected Topics in Signal Processing*, volume 5, no. 6, 2011: pp. 1133–1143.
- [23] Koretz, A.; Tabrikian, J. Maximum a posteriori probability multiple-pitch tracking using the harmonic model. *IEEE Transactions on Audio, Speech, and Language Processing*, volume 19, no. 7, 2011: pp. 2210–2221.
- [24] Smaragdis, P.; Brown, J. C. Non-negative matrix factorization for polyphonic music transcription. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No. 03TH8684)*, IEEE, 2003, pp. 177–180.
- [25] Vincent, E.; Bertin, N.; et al. Adaptive harmonic spectral decomposition for multiple pitch estimation. *IEEE Transactions on Audio, Speech, and Language Processing*, volume 18, no. 3, 2009: pp. 528–537.
- [26] Bertin, N.; Badeau, R.; et al. Enforcing harmonicity and smoothness in Bayesian non-negative matrix factorization applied to polyphonic music transcription. *IEEE Transactions on Audio, Speech, and Language Processing*, volume 18, no. 3, 2010: pp. 538–549.
- [27] Ochiai, K.; Kameoka, H.; et al. Explicit beat structure modeling for non-negative matrix factorization-based multipitch analysis. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2012, pp. 133–136.
- [28] Grindlay, G.; Ellis, D. P. Transcribing multi-instrument polyphonic music with hierarchical eigeninstruments. *IEEE Journal of Selected Topics in Signal Processing*, volume 5, no. 6, 2011: pp. 1159–1169.
- [29] Dessein, A.; Cont, A.; et al. Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence. 2010.

- [30] Bello, J. P.; Daudet, L.; et al. Automatic piano transcription using frequency and time-domain information. *IEEE Transactions on Audio, Speech, and Language Processing*, volume 14, no. 6, 2006: pp. 2242–2251.
- [31] Poliner, G. E.; Ellis, D. P. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, volume 2007, no. 1, 2006: p. 048317.
- [32] Ryyanen, M. P.; Klapuri, A. Polyphonic music transcription using note event modeling. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.*, IEEE, 2005, pp. 319–322.
- [33] Raczyński, S. A.; Ono, N.; et al. Note detection with dynamic bayesian networks as a postanalysis step for NMF-based multiple pitch estimation techniques. In *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, IEEE, 2009, pp. 49–52.
- [34] Gouyon, F.; Dixon, S. A review of automatic rhythm description systems. *Computer music journal*, volume 29, no. 1, 2005: pp. 34–54.
- [35] Desain, P.; Honing, H. Computational models of beat induction: The rule-based approach. *Journal of new music research*, volume 28, no. 1, 1999: pp. 29–42.
- [36] Large, E. W.; Kolen, J. F. Resonance and the perception of musical meter. *Connection science*, volume 6, no. 2-3, 1994: pp. 177–208.
- [37] Dixon, S. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, volume 30, no. 1, 2001: pp. 39–58.
- [38] Böck, S.; Korzeniowski, F.; et al. madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia*, Amsterdam, The Netherlands, 10 2016, pp. 1174–1178, doi:10.1145/2964284.2973795.
- [39] Cemgil, A. T.; Kappen, B. Monte carlo methods for tempo tracking and rhythm quantization. *arXiv preprint arXiv:1106.4863*, 2011.
- [40] Digital Audio Basics: Sample Rate and Bit Depth. 7 2019, [Cited 2019-11-24]. Available from: <https://www.izotope.com/en/learn/digital-audio-basics-sample-rate-and-bit-depth.html>
- [41] Wendt, S. *Roots of Modern Technology: An Elegant Survey of the Basic Mathematical and Scientific Concepts*. Springer Berlin Heidelberg, 2010, ISBN 9783642120626. Available from: <https://books.google.com.ua/books?id=c8TdQmtOD-AC>

-
- [42] Ghosh, S. *Signals and Systems*. Always Learning, Pearson Education Canada, 2005, ISBN 9788177583809. Available from: https://books.google.com.ua/books?id=Sg6y_EBYCEsC
 - [43] Dregvaite, G.; Damasevicius, R. *Information and Software Technologies: 21st International Conference, ICIST 2015, Druskininkai, Lithuania, October 15-16, 2015, Proceedings*. Communications in Computer and Information Science, Springer International Publishing, 2015, ISBN 9783319247700. Available from: <https://books.google.com.ua/books?id=-pLDCgAAQBAJ>
 - [44] Müller, M.; Zunner, T. Envelope and ADSR Model. [Cited 2019-11-28]. Available from: https://www.audiolabs-erlangen.de/resources/MIR/FMP/C1/C1S3_Timbre.html
 - [45] Engel, J.; Resnick, C.; et al. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. 2017, [arXiv:1704.01279](https://arxiv.org/abs/1704.01279).
 - [46] Ingmire, J. Acquiring ‘perfect’ pitch may be possible for some adults. May 2015, [Cited 2019-12-23]. Available from: <https://news.uchicago.edu/story/acquiring-perfect-pitch-may-be-possible-some-adults>
 - [47] Russell, R. Hearing, Columns and Comb Filtering. 2006, [Cited 2019-12-23]. Available from: <http://www.roger-russell.com/columns/combfilter2.htm>
 - [48] Sha’ath, I. 2011, [Cited 2019-12-23]. Available from: <http://www.ibrahimshaath.co.uk/keyfinder/>
 - [49] LilyPond - music notation for everyone. 2019, [Cited 2020-01-05]. Available from: <http://lilypond.org/>
 - [50] and Gribonval, R.; Févotte, C. Performance measurement in blind audio source separation. *IEEE transactions on audio, speech, and language processing*, volume 14, no. 4, 2006: pp. 1462–1469.
 - [51] Stoter, F.-R.; Uhlich, S.; et al. Open-Unmix - A Reference Implementation for Music Source Separation. *Journal of Open Source Software*, 2019, doi:10.21105/joss.01667. Available from: <https://doi.org/10.21105/joss.01667>
 - [52] Defferrard, M.; Benzi, K.; et al. FMA: A Dataset for Music Analysis. In *18th International Society for Music Information Retrieval Conference*, 2017. Available from: <https://arxiv.org/abs/1612.01840>
 - [53] Cancela, P.; Rocamora, M.; et al. An Efficient Multi-Resolution Spectral Transform for Music Analysis. In *ISMIR*, 2009, pp. 309–314.

BIBLIOGRAPHY

- [54] McGrain, M. *Music Notation: Theory and Technique for Music Notation*. Berklee guide, Berklee Press, 1990, ISBN 9780793508471. Available from: https://books.google.cz/books?id=S_y7JAZqx6QC
- [55] Making Music. An Explanation of Clefs: Treble, Bass, Alto, Tenor. [Cited 2019-10-05]. Available from: <https://makingmusicmag.com/explanation-clefs-treble-bass-alto-tenor/>
- [56] Key signature and music staff. 4 2012, [Cited 2019-10-23]. Available from: <https://www.aboutmusictheory.com/key-signature.html>
- [57] Encyclopædia Britannica, inc. Time signature. 11 2017, [Cited 2019-10-23]. Available from: <https://www.britannica.com/art/time-signature>

Acronyms

WAVE	Waveform Audio File Format
LPCM	Linear pulse-code modulation
FFT	Fast Fourier transform
DFT	Discrete Fourier transform
IDFT	Inverse Discrete Fourier transform
CNN	Convolutional Neural Network
SiSeC	Source Separation campaign
AMT	Automatic music transcription
ML	Machine learning
MIREX	Music Information Retrieval Evaluation eXchange
HPS	Harmonic partial sequence
MAP	Maximum a posteriori
EM	Expectation-maximisation
NMF	Non-negative matrix factorisation
HMM	Hidden Markov model
DBN	dynamic Bayesian network
ADSR	attack, decay, sustain, and release
MIDI	Musical Instrument Digital Interface
BPM	beats per minute

A. ACRONYMS

ARIMA Autoregressive integrated moving average

PDF Portable Document Format

CLI command line interface

DNN deep neural network

SDR Source to Distortion Ratio

SIR Source to Interferences Ratio

SNR Sources to Noise Ratio

SAR Sources to Artifacts Ratio

FMA Free Music Archive

MIR Music Information Retrieval

Musical notation

Music notation, when properly applied, can completely describe any musical score in a simple, concise manner. In order to achieve this, music notation must describe all definable parameters of each sound, specifically[54]:

- duration
- pitch
- dynamic
- timbre

Duration is described by time signature ($\frac{4}{4}$, $\frac{3}{4}$, $\frac{7}{8}$, etc.), tempo (primarily, beats per minute: $\text{♩} = 120$), and duration values of note-heads (♩, ♪, ♫, etc.) and rests (—, ♯, ♮, etc.):



Pitch is defined by position of the note on the staff, key, accidentals (♭, ♯, ♮), and the specified clef (♩, ♪, ♫, etc.):



Dynamic of a sound describes its amplitude or loudness (*pp*, *p*, *mf*, *f*, *ff*, etc.), its emotional intensity and change through time.

Timbre describes specific color of a played note/sound. Timber primarily depends on the instrument played but also can define other instrumental directions (i.e. *on bell of cymbal*, etc.)

B.1 The Staff

The base for all musical scores is the *staff*. All other music symbols are placed on the staff or in relation to it.

The staff consists of five horizontal lines and four spaces between the lines. Every note-head is placed on one of the lines or on one of the spaces between the lines. The higher the note-head on the staff - the higher the pitch of the produced note.



B.2 Leger Lines

Obviously, five lines and five spaces can provide only limited range of notes (precisely, eleven places to put the note-head, including just beneath the first(bottom) line and above fifth(top) line). If notes from outside this range are needed, they are placed on or between so-called *Leger lines*. These are the lines placed above or beneath the main staff only in places where they are needed, so for each note individually.



B.3 Clefs

The specified *clef* defines location of each pitch on the staff. The most commonly used clefs are the Treble and the Bass clefs[55].

B.3.1 The Treble Clef

The *Treble Clef* (or *G clef*, because the middle curl of it encircles line on the staff that represents a G-note) is used for most high-sounding instruments (i.e. violin, guitar, ukulele, flute, clarinet, saxophone, trumpet, etc.).



As it defines second line as G, the lines on the staff, from bottom to top, are E, G, B, D, F. The spaces then are F, A, C, E. The middle C¹⁰ goes on the first leger line below the treble staff.

B.3.2 The Bass Clef

The *Base Clef* (or *F clef*, because line between two dots on the symbol represents an F-note) is used for low sounding instruments (i.e. bass guitar, cello, trombone, tuba, etc.)



As it defines fourth line as F, the lines on the staff are G, B, D, F, A, and the spaces are A, C, E, G. The middle C goes on the first leger line above the bass clef.

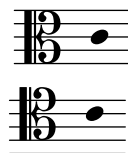
B.3.3 The Percussion Clef

The *Percussion Clef* is commonly used for drum-set notation. Each line and space represent different part of the drum kit. They are often predefined at the start of the part in so-called *key* or *legend*, or when they first appear in the score.



B.3.4 The Alto and Tenor Clefs

Alto Clef (or *C clef*, because line in the middle of the alto staff represent middle C) and The *Tenor Clef* are less often used clefs. The viola and the alto trombone are generally the only instruments that use the Alto clef. Tenor clef is occasionally used to represent the upper ranges of the cello, double bass, bassoon, and trombone.

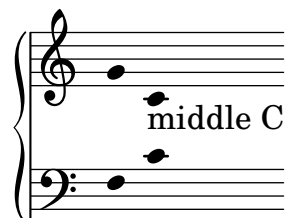


The lines of the alto staff are F, A, C, E, G, and the spaces are G, B, D, F. Similarly, for tenor clef, C is moved up one line from alto clef, making the notes on the lines D, F, A, C, E and notes in the spaces E, G, B, D.

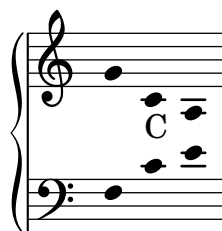
¹⁰*Middle C* is a commonly used reference note. It is a closest C to the middle of a standard 88 key piano (specifically, fourth C from the left). It is around 261.63 hertz.

B.3.5 The Great Staff

The *Great Staff* or the *Grand Staff* is a combination of the treble staff and the bass staff. Usually used by piano or harp musicians.

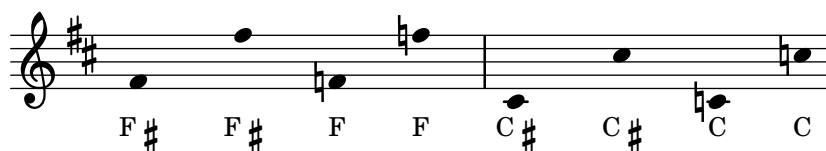


Often they also divide score into to parts played by left and right hand (i.e. on piano, treble clef part with the right hand, bass clef part with left hand). So, even if some notes belong to treble clef they may be put on leger lines above bass clef if played by left hand and vice versa.



B.3.6 Key signature

Key signature is a series of sharp symbols or flat symbols placed on the staff, designating notes that are to be consistently played one semitone higher or lower than the equivalent natural notes (for example, the white notes on a piano keyboard) unless otherwise altered with an accidental. Key signatures are generally written immediately after the clef at the beginning of a line of musical notation, although they can appear in other parts of a score, notably after a double bar[56].



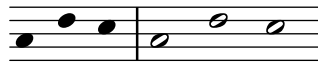
Key *D major* (defined in example above) consists of notes D, E, F#, G, A, B, C#. So, after the clef, notes F and C marked with a #, so, when they occur in a score without any accidentals, they are played one semitone higher (C# instead of C, etc.)

B.4 Rhythmic Description

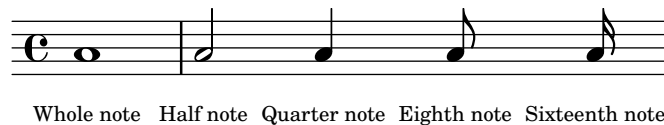
Alongside with pitch, it is required to describe rhythm. *Rhythmic description* determines exactly when note should be played and when it should stop playing. Notationally it is defined by note-heads, stems, flags, beams, rests, and time signature.

B.4.1 Note-heads, stems, flags, beams

There are two types of note heads open and closed.



Stems are vertical lines attached to the side of the notes-head. Together with flags, beams, and augmentation dots they define duration value:



Two half note have the same duration as one whole note, two quarter notes have the same duration as one half note and so on.

B.4.2 Rests

Same as for notes, we can define pauses in music - *rests*:



Whole rest, half rest, quarter rest, and so on accordingly.

B.4.3 Time signatures

Time signature is a sign that indicates the metre of a composition. Most time signatures consist of two vertically aligned numbers, such as $\frac{2}{2}$, $\frac{3}{4}$, $\frac{6}{8}$, and $\frac{11}{16}$. The top figure reflects the number of beats in each measure, or metrical unit; the bottom figure indicates the note value that receives one beat (here, respectively, half note, quarter note, eighth note, and sixteenth note). When measures contain an uneven number of beats falling regularly into two subgroups, the division may be indicated as, for instance, $\frac{3+4}{4}$ instead of $\frac{7}{4}$ [57].

B. MUSICAL NOTATION



$\frac{4}{4}$ is such a common time signature that sometimes it is specified with **C** and $\frac{2}{2}$ as **♩**.

Contents of enclosed CD

	readme.txt	the file with CD contents description
	src	the directory of source codes
	mimt	implementation sources
	thesis	the directory of L ^A T _E X source codes of the thesis
	text	the thesis text directory
	thesis.pdf	the thesis text in PDF format