UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

AUTOMATIC TRANSCRIPTION FOR POLYPHONIC MUSIC

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Juan Martín Valero

In partial fulfilment

of the requirements for the degree in

AUDIOVISUAL SYSTEMS ENGINEERING

Advisor: Antonio Bonafonte

Barcelona, May 2016

Abstract

We are living times where technology and music go hand in hand, and everyday more musicians and producers integrate new music software into their workflow.

This project is the study and development of a first prototype for a complete automatic transcription system for polyphonic music, which is able to generate a MIDI file from an audio signal. This first prototype is focused on piano music.

The developed algorithm combines different techniques to put together the six different blocks that form the final system. The core of the project is the multiple fundamental frequency estimation and it is based on the "Iterative Estimation and Cancellation" method, initially proposed by Anssi Klapuri. The general system combines this multi F0 estimation method with techniques from other research papers and some new methods proposed in this work.

With an accuracy of 0.7402 for melodies and 0.6142 for low polyphonic recordings, we can say that the results are good for simple scenarios.

Resum

Estem vivint uns temps en els quals la tecnologia i la música van de la mà, i cada dia més músics i productors estan integrant nou software musical a la seva manera de treballar.

Aquest projecte és l'estudi i el desenvolupament d'un primer prototip per a un sistema complet de transcripció musical automàtic per a música polifònica, que és capaç de generar un fitxer MIDI a partir d'un senyal d'àudio. Aquest primer prototip se centra en la música de piano.

L'algoritme desenvolupat combina diferents tècniques per a crear els sis blocs que formen el sistema final. El nucli del projecte és l'estimació de múltiples freqüències fonamentals i es basa en el mètode de "Estimació i Cancel·lació Iterativa", proposat inicialment per Anssi Klapuri. El sistema general combina aquest mètode d'estimació de multiples F0 amb tècniques d'altres publicacions i alguns nous mètodes proposats en aquest treball.

Amb una *accuracy* de 0.7402 per a melodies i de 0.6142 per a gravacions amb baixa polifonia, podem dir que els resultats obtinguts són bons per a casos simples.

Resumen

Vivimos tiempos en los que la tecnología y la música van de la mano, y cada día más músicos y productores integran nuevo software musical en su forma de trabajar.

Este proyecto es el estudio y desarrollo de un primer prototipo para un sistema completo de transcripción musical automática para música polifónica, que es capaz de generar un fichero MIDI a partir de una señal de audio. Este primer prototipo se centra en la música de piano.

El algoritmo desarrollado combina diferentes técnicas para crear los seis bloques que forman el sistema final. El núcleo del proyecto es la estimación de múltiples frecuencias fundamentales y se basa en el método de "Estimación y Cancelación Iterativa", propuesto inicialmente por Anssi Klapuri. El sistema general combina este método de estimación de múltiples F0 con técnicas de otras publicaciones y algunos nuevos métodos propuestos en este trabajo.

Con una *accuracy* de 0.7402 para melodías y de 0.6142 para grabaciones con baja polifonía, podemos decir que los resultados obtenidos son buenos para casos simples.

To my brother and my mother.

To Ignasi, Dani and everyone who has made these last years count. You know who you are.

Acknowledgements

I would like to thank my project advisor, Antonio Bonafonte, for all the advice and support given during all these months, I know it has not been easy.

.

Revision history and approval record

Revision	Date	Purpose
0	30/03/2016	Document creation
1	05/05/2016	Document revision
2	11/05/2016	Document revision
3	13/05/2016	Document approval

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Juan Martín Valero	jmartinvalero@hotmail.com
Antonio Bonafonte	antonio.bonafonte@upc.edu

Written by:		Reviewed and approved by:	
Date	30/03/2016	Date	13/05/2016
Name	Juan Martín Valero	Name	Antonio Bonafonte
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables:

1. Introduction

The way music is produced and recorded has changed drastically this last decade. Nowadays musicians don't need to rely any more on expensive studios to record their music, because anyone can set up a home studio for around 300 €. One of the tools that have had a major impact in this evolution is the virtual instrument, a piece of software that simulates the sound of any acoustic instrument or synthesizer and allows the musician to play the instrument without physically having it or even knowing how to play it, just using a MIDI controller or a mouse.

With this project my intention was to change the way musicians use virtual instruments. The usual way to play these virtual instruments, as mentioned above, is by using an external MIDI controller. These MIDI controllers are keyboards without integrated sounds, they can only play a sound through a computer's virtual instrument. They are a great tool and very useful, but requires this piece of hardware (the keyboard) and some knowledge on how to play it, which are both economic and technical limitations.

So one thought came to me: "If one is a musician, he or she can already play an instrument or sing. It would be great if they could use the abilities and resources they already have in order to play virtual instruments".

The whole purpose of this project, then, as a final product would be to provide musicians with a tool that allows them to use any instrument, including vocals, to play virtual instruments. This is achieved by converting the acoustic signal into MIDI notation, which can afterwards be edited or played with any virtual instrument. The developed system in this project is a first prototype of this idea and it is focused on piano music.

In this document I intend to explain as comprehensive as possible the structure and functions of all the blocks that form the developed system.

1.1. Objectives

The goal of this project is to study the concept of automatic music transcription and then design and implement an algorithm to convert an audio signal into a MIDI file that defines the following MIDI parameters: note number, note onset, note offset.

1.2. Requirements and specifications

Project requirements

- Design and develop a program to convert an audio input signal into a MIDI file.
- Minimize the error in the following MIDI parameters: note number, note offset and note duration.
- Work with both monophonic and polyphonic audio.
- Build an evaluation system with a complete database to test the algorithms and compare results.

Project specifications

- Developed in MATLAB.
- Main focus on piano music.
- Let the user choose several options, if desired, like the type of input audio (monophonic or polyphonic) or fix the polyphony number to improve the accuracy.

1.3. Methods and procedures

This project starts from scratch, it is not a continuation of another project.

All the code has been developed by me except the MIDI generation algorithm from 3.7, which was developed by Ken Schutte in 2009 [19].

1.4. Work plan

1.4.1. Work packages, tasks and milestones

Work packages

WP1: Learn about VST and available software.

WP2: Research on state of the art techniques.

WP3: Transcription algorithm design and implementation.

WP4: Algorithm for MIDI generation.

WP5: Final report and oral defense.

Tasks and milestones

WP#	Task#	Short title	Milestone	Date
1	1	Learn about VST technology	Understand the VST format	30/09/2015
1	2	Learn about VST plugin development	Develop a few simple VST plugins	30/09/2015
1	3	Look into existing software	Get an idea of the available software	15/09/2015
1	4	Learn about MIDI creation	Create a few simple MIDI files	07/10/2015
2	1	Research on music transcription techniques	Selection of interesting techniques	18/10/2015
3	1	Multi F0 algorithm design	Design	21/11/2016
3	2	Multi F0 algorithm implementation	MATLAB program	21/03/2016
3	3	Onset detection algorithm design	Design	20/01/2016
3	4	Onset detection algorithm implementation	MATLAB program	27/01/2016
3	5	Tracking algorithm design	Design	24/02/2016
3	6	Tracking algorithm implementation	MATLAB program	21/03/2016
3	7	Testing	Results	21/03/2016
4	1	Integrate MIDI generation algorithm	MATLAB program	30/03/2016
4	2	Testing	Results	30/03/2016
5	1	Write the final report	Final report document	27/04/2016
5	2	Prepare the presentation	Presentation slides	10/05/2016

Table 1: Tasks and milestones

1.4.2 Gantt diagram

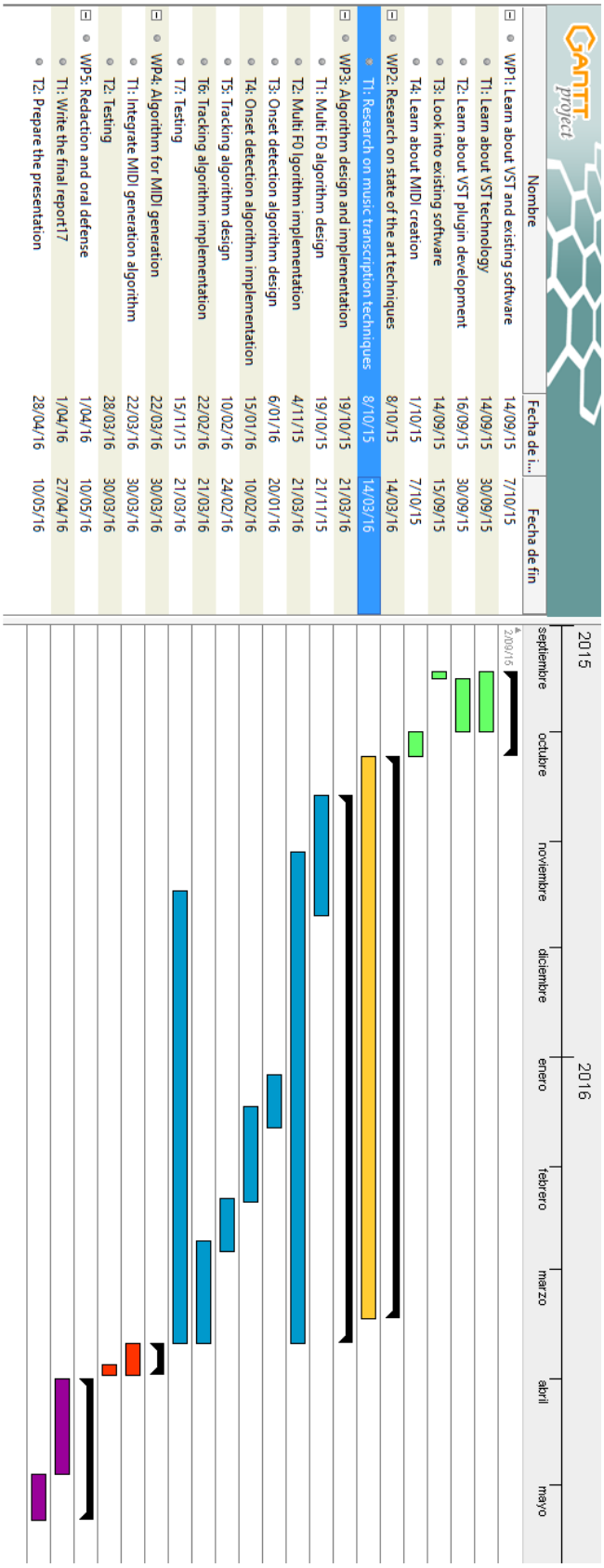


Figure 1: Gantt diagram

1.5. Thesis structure

The rest of the document is organized as follows:

- *Chapter 2: Automatic Music Transcription: What is it and where are we today?*

  Includes a general background on Automatic Music Transcription and MIDI, a review on the most recent competition and the leading commercial products.

- *Chapter 3: Multiple Fundamental Frequency Estimation*

  Detailed explanation of the system's core, the Multi F0 Estimation algorithm.

- *Chapter 4:* The complete music transcription system

  Based on the multiple F0 estimation from the previous chapter, a transcription system is proposed that includes tracking, onset detection and MIDI generation.

- *Chapter 5: Evaluation of the Transcription System*

  Includes the obtained results from the system's evaluation.

- *Chapter 6: Budget*

  Includes an estimated budget for the project.

- *Chapter 7: Conclusions and future development*

  Final thoughts on the project and future development.

- *Bibliography*

- *Glossary*

2. Automatic Music Transcription: What is it and where are we today?

This chapter includes a general background on Automatic Music Transcription and MIDI, a review on the most recent competition and the commercial products that are leading the industry right now.

2.1. Music transcription and MIDI

Generally speaking, music transcription means notating a piece of music in a way that can later be understood by someone of something. There are two types of music transcription: Manual and automatic. The first is usually done by musicians and transcribed into a score, while the second is done by computers and usually transcribed into MIDI notation.

MIDI (Musical Instrument Digital Interface) is a standard that saw the light in 1987 and allows a wide variety of electronic musical instruments, computers and other related devices to communicate with one another. MIDI is a format that carries events with information on a set of parameters needed to define a music note, such as note number (pitch), note onset, note offset, velocity, vibrato, modulation, sustain, panning, channel and also clock signals to allow tempo synchronization between devices (for example, to synchronize two synthesizers). Since these are general parameters, the musical information carried by a MIDI file can be played with any virtual instrument, which gives a lot of versatility to the musician. In addition, it can also be modified within a MIDI editor.

The figure below shows the classical piano roll representation of a MIDI file containing 7 chords.

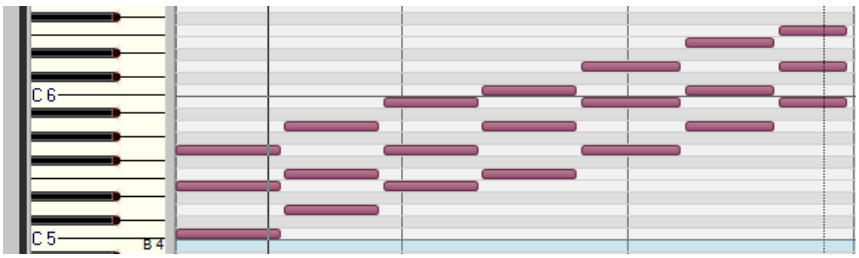


Figure 2: Piano roll representation of a MIDI file

The concept of automatic music transcription goes back to 1977, when it was first used by the researchers James A. Moorer, Martin Piszczalski and Bernard Galler. They believed that a computer could be programmed to detect pitches of melody lines, chords patterns and rythms. This is not a simple goal but has encouraged researchers for almost 40 years now and is still a very active field for research. Through the years, the music recognition field has recycled many techniques originally developed for speech recognition, which has seen a better financial support.

2.2. Main tasks and approaches

Automatic music transcription is a very complex field and it's composed by several tasks like multiple fundamental frequency estimation, onset and offset detection and note tracking.

These tasks can be approached mainly in two ways: Pure audio processing or machine learning. There are great research papers on both approaches and they both can give good results. In fact, the best approach is usually a combination of the two. In this thesis we will be focusing on the first one, audio processing.

The method and the algorithm chosen will depend highly on the final goal of the application. For instance, if the desired goal is to do a real time transcription, the main focus will be the speed of the algorithm. However, if the time is not the main issue, the system could use more complex algorithms in order to give a more accurate result.

2.3. Competitions

There is an annual MIREX (Music Information Retrieval Evaluation eXchange) [21] competition which serves as a reference point for researchers and lets them evaluate their algorithms and compare them to the rest of the contestants.

The MIREX competition related to this topic is divided in two different tasks: Multiple Fundamental Frequency Estimation and Tracking.

The results for last year's competition (MIREX 2015) are shown below and have been taken from the official MIREX website [21]. These are the results evaluating the algorithms with the MIREX dataset. The different evaluation metrics used in this competition are defined as follows:

$$Precision = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t)+FP(t)} \quad Recall = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t)+FN(t)} \quad Accuracy = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t)+FP(t)+FN(t)}$$

$$F\text{-}measure = \frac{2 \times precision \times recall}{precision+recall}$$

$TP \stackrel{def}{=}$ True Positive = Number of F0s that correctly correspond between the ground-truth F0 and the reported F0 set for that frame.

$FP \stackrel{def}{=}$ False Positive = Number of F0s detected that do not exist in the ground-truth set for that frame.

$FN \stackrel{def}{=}$ False Negative = Difference between the number of reported negatives at frame t and the number of negatives in the ground-truth at frame t.

$$E_{tot} = \frac{\sum_{t=1}^{T} \max\left(N_{ref}(t), N_{sys}(t)\right) - N_{corr}(t)}{\sum_{t=1}^{T} N_{ref}(t)} \qquad E_{sub} = \frac{\sum_{t=1}^{T} \min\left(N_{ref}(t), N_{sys}(t)\right) - N_{corr}(t)}{\sum_{t=1}^{T} N_{ref}(t)}$$

$$E_{miss} = \frac{\sum_{t=1}^{T} \max(0, N_{ref}(t) - N_{sys}(t))}{\sum_{t=1}^{T} N_{ref}(t)} \qquad E_{fa} = \frac{\sum_{t=1}^{T} \max(0, N_{sys}(t) - N_{ref}(t))}{\sum_{t=1}^{T} N_{ref}(t)}$$

$E_{tot} \stackrel{def}{=}$ Total error. It is not necessarily bounded by 1.

$E_{sub} \stackrel{def}{=}$ Substitution error = Counts the number of ground-truth F0s for each frame that were not returned, but some other incorrect F0s were returned instead. It is bounded between 0 and 1.

$E_{miss} \stackrel{def}{=}$ Missed errors. It is bounded between 0 and 1.

N_{ref} = Number of F0s in the ground-truth list for frame t.

N_{sys} = Number of reported F0s.

N_{corr} = Number of correct F0s for that frame.

2.3.1. Multi F0 Estimation

This task is evaluated comparing the estimated F0s frame by frame with the ground truth. Table 2 shows the average score for the evaluation, done with 40 test files coming from 3 different sources: woodwind quintet recording of bassoon, clarinet, horn, flute and oboe; Rendered MIDI using RWC database and a quartet recording of bassoon, clarinet, violin and saxophone. These files range from polyphony 2 to 5.

The first column on the table is the code reference to the different algorithms submitted.

	Precision	*Recall*	*Accuracy*	*Etot*	*Esub*	*Emiss*	*Efa*
BW1	0.752	0.755	0.654	0.409	0.096	0.149	0.164
CB1	0.804	0.519	0.498	0.529	0.093	0.389	0.047
CB2	0.655	0.460	0.420	0.636	0.174	0.366	0.095
SY1	0.637	0.775	0.588	0.581	0.137	0.088	0.357
SY2	0.640	0.767	0.584	0.571	0.129	0.104	0.338
SY3	0.631	0.749	0.571	0.603	0.146	0.105	0.352
SY4	0.644	0.719	0.567	0.571	0.140	0.141	0.290

Table 2: MIREX 2015 Multi F0 Estimation results

The best result has an accuracy of 0.654. This is far from the ideal score 1, but right now it is a very good score for the task, considering that the database used is very diverse and has high polyphony, which increases the difficulty.

2.3.2. Note tracking

This subtask is evaluated in two different ways. In the first setup, a returned note is assumed correct if its onset is within +/-50ms of a ref note and its F0 is within +/- a quarter tone of the corresponding reference note, ignoring the returned offset values. In the second setup, on top of the above requirements, a correct returned note is required to have an offset value within 20% of the ref notes duration around the ref note's offset, or within 50ms whichever is larger.

A total of 34 files were used in this subtask: 16 from woodwind recording, 8 from IAL quintet recording and 6 piano.

- **Setup 1: Onset only**

	Precision	*Recall*	*Ave. F-measure*
BW2	0.566	0.667	0.601
BW3	0.499	0.618	0.541
CB1	0.527	0.491	0.503
CB2	0.376	0.401	0.374
SY1	0.394	0.639	0.479
SY2	0.372	0.642	0.460

SY3	0.358	0.703	0.462
SY4	0.354	0.691	0.455

Table 3: MIREX 2015 note tracking by onset only results

- **Setup 2: Onset and offset**

	Precision	***Recall***	***Ave. F-measure***
BW2	0.329	0.405	0.356
BW3	0.280	0.366	0.311
CB1	0.315	0.304	0.306
CB2	0.201	0.228	0.206
SY1	0.254	0.430	0.315
SY2	0.232	0.424	0.294
SY3	0.218	0.441	0.285
SY4	0.207	0.420	0.271

Table 4: MIREX 2015 note tracking

The best results are an Ave. F-measure of 0.601 (onset only) and 0.356 (onset and offset). As expected, these results are lower than the accuracy in the Multi F0 Estimation task, since note tracking involves F0 estimation too. The scores get pretty low when combining different evaluations (F0 estimation, onset and offset), so we can see from these results that automatic music transcription is not something trivial.

- **Note tracking piano only: Onset and offset**

These next results are based on the same note tracking task and evaluation but using only piano recordings. 6 piano recordings are used to evaluate this subtask.

	Precision	***Recall***	***Ave. F-measure***
BW2	0.204	0.203	0.203
BW3	0.239	0.228	0.233
CB1	0.276	0.212	0.239
CB2	0.208	0.152	0.174
SY1	0.148	0.175	0.157
SY2	0.105	0.144	0.119
SY3	0.179	0.219	0.193
SY4	0.136	0.188	0.156

Table 5: MIREX 2015 note tracking piano only results

- **Note tracking piano only: Onset only**

	Precision	*Recall*	*Ave. F-measure*
BW2	0.640	0.643	0.641
BW3	0.704	0.683	0.692
CB1	0.743	0.611	0.667
CB2	0.559	0.450	0.494
SY1	0.438	0.562	0.480
SY2	0.368	0.540	0.429
SY3	0.476	0.643	0.533
SY4	0.411	0.630	0.487

Table 6: MIREX 2015 note tracking piano only (onset only)

Observing now the piano only results, we see that the score is higher for onset only (0.692) but lower for onset and offset (0.239). Although the database used is not very extensive, this appears to be a result of the attack and release times of the instruments involved. In the first case, it is mostly wind instruments, which have higher attack time (the onset more diffuse) and lower release time (the offset is more clear). However, in the second case it is piano only, which has lower attack time (the onset if more clear) and higher release time (the offset is more diffuse). This is why the piano-only evaluation has better results with onset only and worse results with onset and offset.

2.4. Commercial products

There are many commercial software out there that performs the music transcription task. The most famous and most common used by professionals in recording studios is Melodyne [12], a software developed by Celemony. It can run both as a standalone application and as a VST plugin, and although it's primarily used for fine tuning voices and instruments it can also export the conversion to a MIDI file. It has three algorithms to choose from depending on the type of audio that you want to analyse: Melodic, polyphonic and percussive.

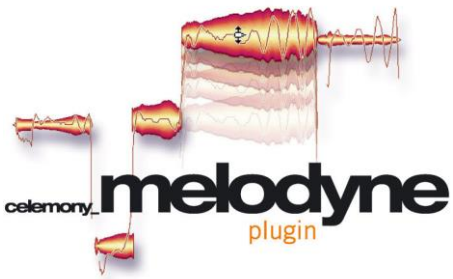


Figure 3: Melodyne logo

A part from this, there are also some DAWs that have implemented in their latest versions an integrated functionality for audio to MIDI conversion. This is great because it's perfectly integrated within the DAW, so usually it's easiest and fastest to use. The downside, however, is that these algorithms are less mature than melodyne. They are much newer and they are developed by companies that, unlike Celemony, are not specialised in this kind of software. Also, since they are integrated, they can only be used inside that particular DAW, so that can be a downside to a user who has preference for another DAW that doesn't have this functionality.

Some of these DAWs are Ableton Live, Reaper, Cubase or Sonar.

Figure 4: Ableton, Reaper and Cubase logos respectively

- **Free software alternatives**

  There are also some free software alternatives, although the quality is not as good as the previous ones. WIDI, by Widisoft, would be a good example.

Now that we have a general background on the topic, let's get into the details of the implemented algorithms in chapters 3 and 4.

3. Multiple Fundamental Frequency Estimation

This block is the core of the system and it is the most complex one. It is based on a paper published by Anssi Klapuri in 2006 called "*Multiple Fundamental Frequency Estimation by Summing Harmonic Amplitudes*" [1], the core of it being the *"Iterative estimation & cancellation"* method. The goal of this block is to estimate the fundamental frequencies F0s on every frame and give the result in MIDI note number.

The diagram of the Multi F0 estimation block is shown below in Figure 5.

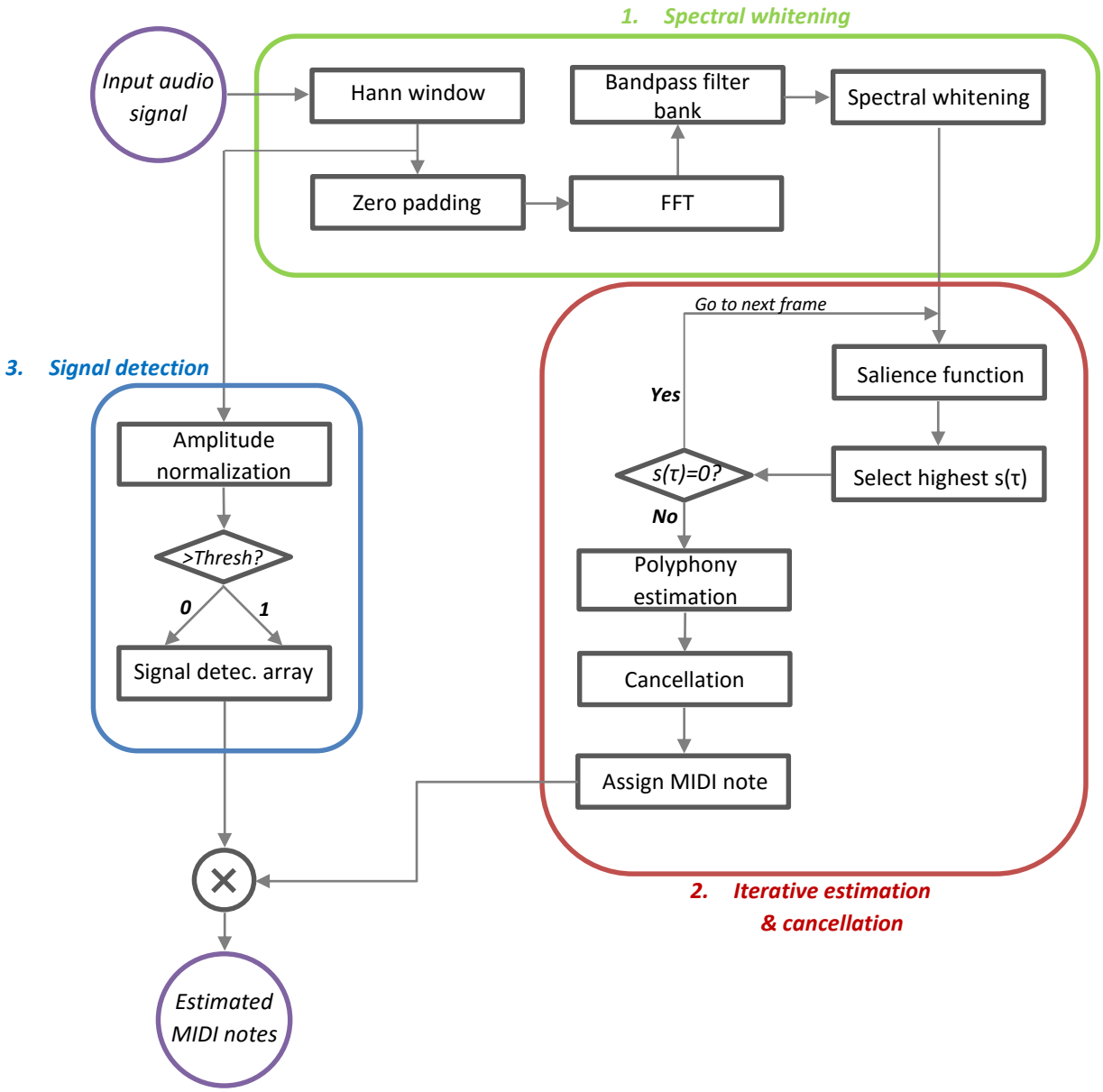


Figure 5: Multi F0 Estimation block diagram

The rest of the chapter will explain in detail the three sub-blocks that compose the Multi F0 Estimation block, which are:

1. Spectral whitening
2. Iterative estimation and cancellation
3. Signal detection

3.1. Spectral whitening

The first sub-block of the Multi F0 estimation is the spectral whitening. Its goal is to flatten the spectrum in order to make the harmonics more relevant in the later processing stages.

1. We start by applying a 92 ms Hann window with a 46 ms hop size to the input audio signal to divide the signal into frames. Next apply a zero padding to 4 times its size, to increase the frequency resolution of the F0 estimation, and then perform a Fast Fourier Transform to each frame.
2. The resulting transformed signal is passed through a bandpass filterbank of 30 filters with its center frequencies located at:

$$c_b = 229 \cdot (10^{(b+1)/21.4} - 1)$$

Each subband b has a triangular response $H_b(k)$ that extends from c_{b-1} to c_{b+1} and is zero elsewhere (see Figure 6).

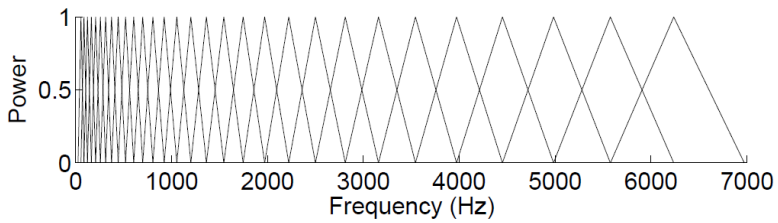


Figure 6: triangular response $H_b(k)$

3. For each subband we compute the average power σ_b and the bandwise compression coefficients γ_b .

$$\sigma_b = \left(\frac{1}{K} \sum_k H_b(k) |X(k)|^2 \right)^{1/2}$$

Where K is the length of the FFT.

$$\gamma_b = {\sigma_b}^{v-1}$$

Where v=0.33 is a parameter determining the amount of whitening applied.

4. The coefficients γ_b are linearly interpolated between the center frequencies c_b to obtain compression coefficients $\gamma(k)$ for all frequency bins k.

5. Finally, the whitened spectrum $Y(k)$ is obtained by weighting the spectrum of the input signal by the compression coefficients:

$$Y(k) = \gamma(k) \cdot X(k)$$

The two figures below show the spectrum of one same frame before and after the whitening. We can observed how the whitened spectrum is more flat than the non-whitened. This is done to give more importance to all the harmonics.

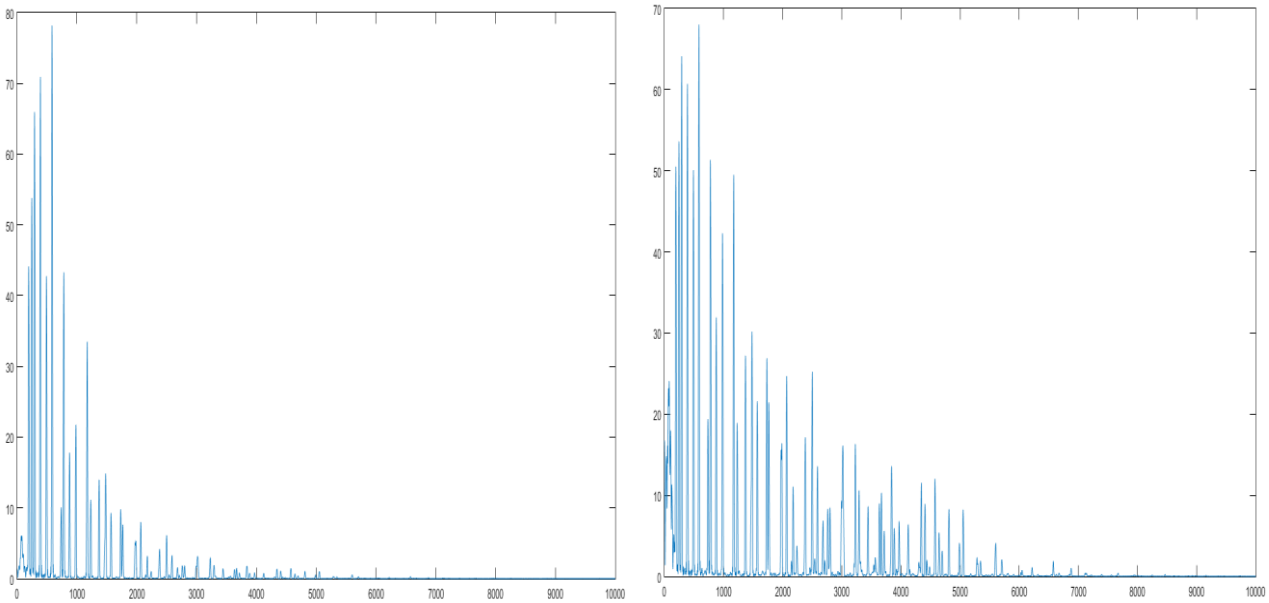


Figure 7: Spectrum before and after the spectral whitening respectively

3.2. Iterative estimation & cancellation

The second sub-block is the iterative estimation & cancellation. Its goal is to estimate all the F0s in every frame. To do that, it goes through every frame and, in each of them, iteratively estimates the first F0, cancels it, estimates the second F0, cancels it and so on until the polyphony detector stops the iterations and moves to the next frame. The cancellation is done to reduce the error of detecting duplicated F0s.

1. For each frame we have to estimate F0. We do that by computing the salience function $s(\tau)$ for all F0 candidates and choosing the one with the highest value.

   The salience function is the sum of all the harmonics weighted by $g(\tau, m)$. For each F0 candidate we accumulate the power of the M harmonics using the weight factor $g(\tau, m)$:

$$s(\tau) = \sum_{m=1}^{M} g(\tau, m) \cdot \max_{k \in k_{\tau,m}} |Y(k)|$$

Where
- $M = 10$ $(number\ of\ harmonics\ used)$.
- $\tau_{min} \leq \tau \leq \tau_{max}$ $where\ F0\ is\ related\ to\ \tau\ through\ F0 = \frac{fs}{\tau}$ $and\ \tau_{min}\ and\ \tau_{max}\ are\ chosen\ in\ relation\ to\ the\ range\ of\ F0$.
- $k_{\tau,m} = \left[\langle\frac{mK}{\tau+\frac{1}{4}}\rangle, \ldots, \langle\frac{mK}{\tau-\frac{1}{4}}\rangle\right]$ $is\ the\ interval\ that\ contains\ the\ surroundings\ of\ the\ harmonics'central\ frequency\ (the\ whole\ peak)$.
- $g(\tau, m) = \frac{\frac{f_s}{\tau} + \alpha}{\frac{m \cdot f_s}{\tau} + \beta}$ $where\ \alpha = 52Hz, \beta = 320Hz$

Figure 8 shows the weight applied to the m[th] harmonic, $g(\tau, m)$, for two different values of τ: $\tau = 220.5$ $(200\ Hz)$ and $\tau = 14.7$ $(3000\ Hz)$.

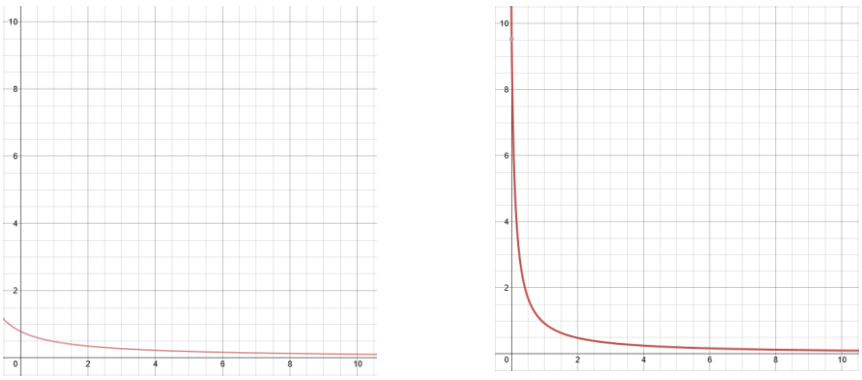


Figure 8: $g(\tau, m)$ for values $\tau = 220.5$ and $\tau = 14.7$ respectively

2. Now that we have computed $s(\tau)$ for all the F0 candidates, we look for the candidate with the highest $s(\tau)$.
   - If the highest value is zero it means there is no signal in this frame, so we assign 0 to the note number and move to the next frame. Under normal conditions, $s(\tau)$ will never be zero because it is very difficult to have absolute silence (no signal) in a recording, but there are some situations where this can happen, for example in audios generated from a MIDI file or in recordings that have been processed with a noise removal algorithm.
   - If the highest value, however, is greater than zero, we check for the polyphony. This means that we have to estimate if this note that we have found is actually a note and not noise. This polyphony estimation is what will make our system stop the iterations and stop detecting notes in a frame.

The condition that will make the system stop the iterations is when $newS \leq oldS$.

Where
- $newS = sAcum/(f^{0.7})$
- $sAcum$: The sum of the $s(\tau)$ from all the estimated notes in the current frame (including the current iteration).
- f: The number of iterations (also known as the polyphony number)
- $oldS$: It is updated on every iteration, and it is equal to the $newS$ from last iteration.

The idea behind this this stopping condition is that we will accept new notes in a frame as long as their $s(\tau)$ is high enough to consider them a valid note. Their $s(\tau)$ will be considered high enough if their contribution to the accumulated $sAcum$ makes $sAcum$ grow at a faster pace than $x^{0.7}$ (shown in figure 9).

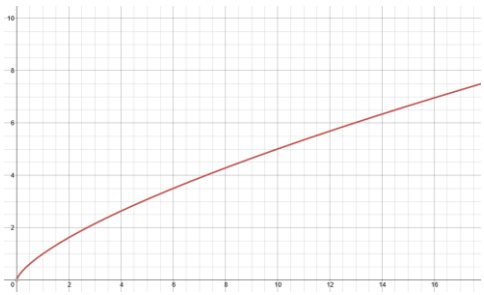


Figure 9: Graph of the equation $x^{0.7}$

3. Once we have estimated the F0, we proceed with the cancellation. The cancellation is done by removing the estimated F0, its harmonics and their surroundings from the original signal, obtaining a remaining signal that will be used for the estimation of the next F0 in the current frame. By performing this cancellation we are reducing the chances of having a very common error which is to detect the same F0 more than once in the same frame.

   This cancellation differs from the one in the original paper, where they propose to do a partial cancellation instead of a complete one, but after implementing and testing both of them, the results obtained were better with a complete cancellation.

4. Finally, we compute the MIDI note number from the F0 value with the equation below:

$$MIDInoteNumber = round\left(12 \cdot \log\left(\frac{F0}{440}\right) + 69\right)$$

3.3. Signal detection

The third and last sub-block is a signal detector. Its goal is to process the frames and decide if there is signal or not.

1. First we apply an amplitude normalization to the whole audio signal to have a maximum amplitude of 1.

2. Then we compare the maximum amplitude value in the frame with a threshold, which in this case is set to 0.005. If the amplitude is below the threshold the system returns 0 and if it's above the threshold it returns 1.
3. With these ones and zeros we create an array which will be multiplied with the array of estimated MIDI notes.

Figure 10 shows the output of the Multi F0 Estimation block when the input is an audio recording of the major triads C, D, E, F, G, A, B. The goal would be to obtain clear horizontal lines between the onsets and offsets of every one of the notes, at the corresponding MIDI note number (Y axis).

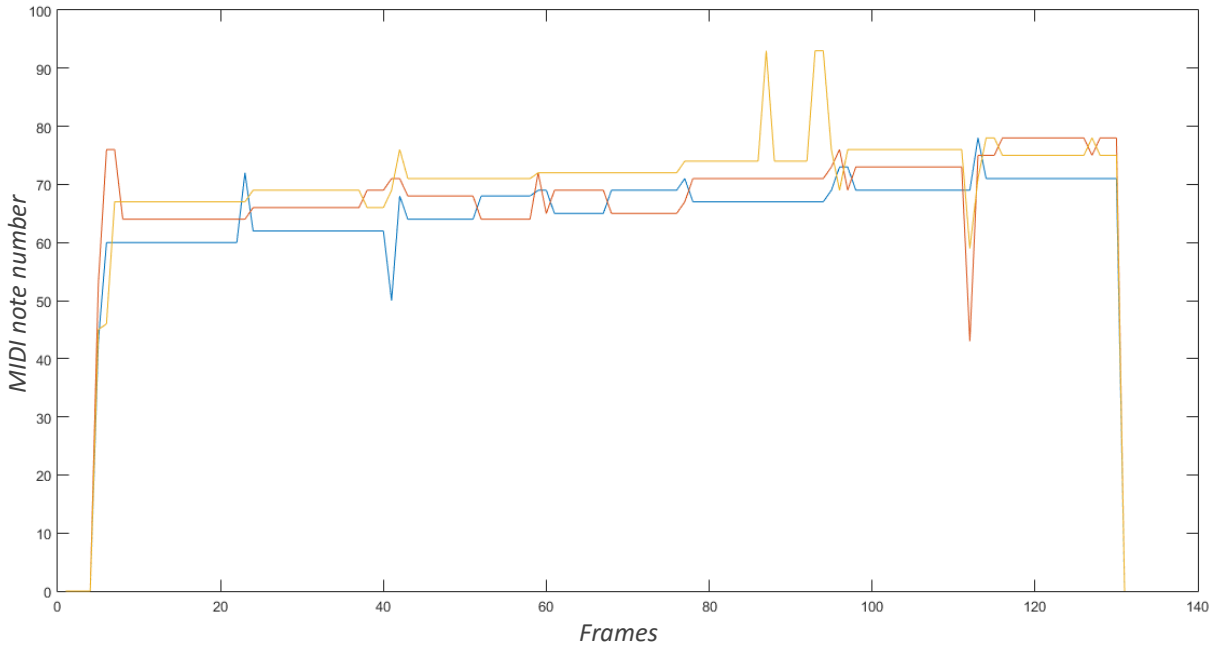


Figure 10: Output of the Multi F0 Estimation block

As we can see, at this early stage the output is still very messy. In chapter 4 we will explain the upcoming blocks, which will process this data and deliver clean notes with well-defined onsets and offsets.

4. The complete music transcription system

In chapter 3 we have seen the Multi F0 Estimation block. Now we will analyse the complete system, which is formed by 6 different blocks:

1. **Multi F0 estimation:** Estimates the F0s present in every frame.
2. **Voice tracking:** Joins the notes that belong to the same voice.
3. **Onset detection:** Detects the beginning of the notes.
4. **Onset processing**: Cleans the output from the voice tracking by applying the onset detection.
5. **Note Tracking:** Provides the essential information of all individual notes: note number, note onset and note offset.
6. **MIDI generation:** Generates the final MIDI file with the information from the note tracking block.

4.1. General diagram

Figure 11 shows the block diagram of the complete system. Here we can get a general idea of how the system is structured and where each block is located in the chain.

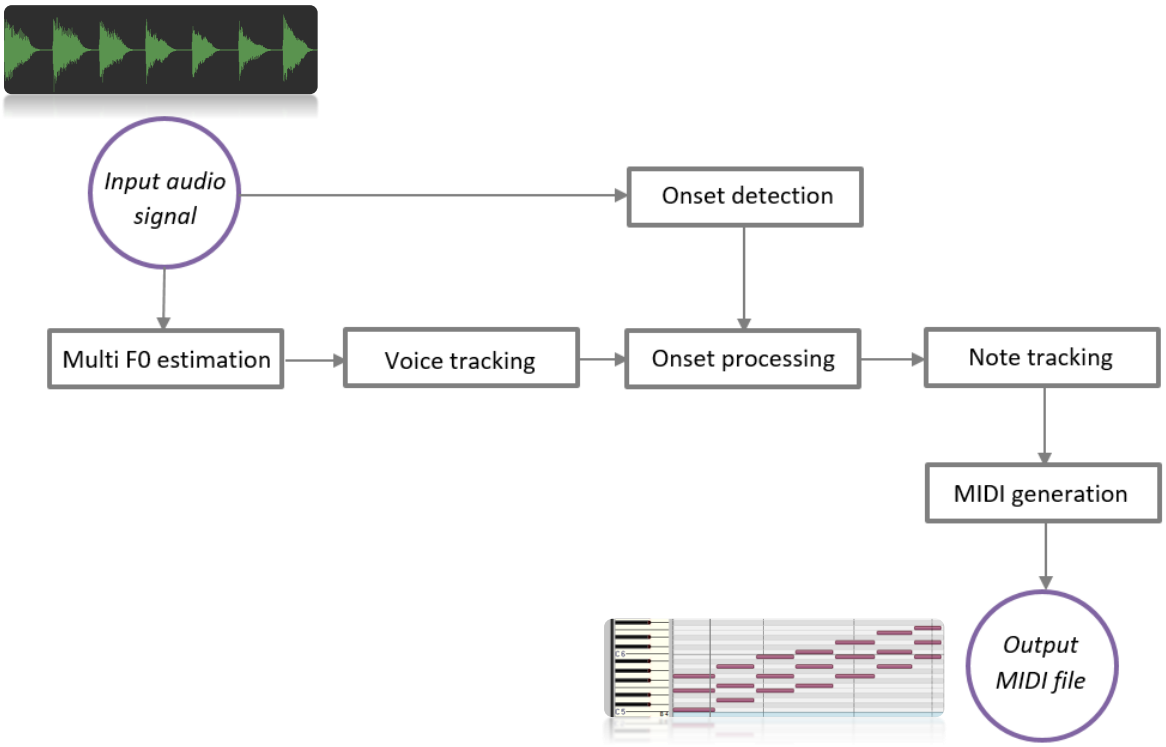


Figure 11: General diagram

We have already seen how the Multi F0 Estimation works. In the following pages we will break down the system and analyse the rest of the blocks one by one alongside a real example, looking at the output signal from every block. That will be useful for a better understanding of the process. *Figure 6* shows the waveform of the input signal that we will use. It contains 7 triads (3 note chords) which correspond to the major chords C, D, E, F, G, A, B:

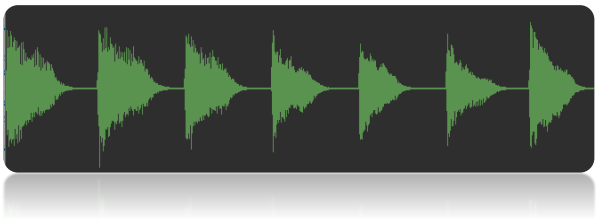


Figure 12: Input audio signal used

As a reminder, figure 13 shows the output from the first block Multi F0 Estimation, using this audio signal as an input.

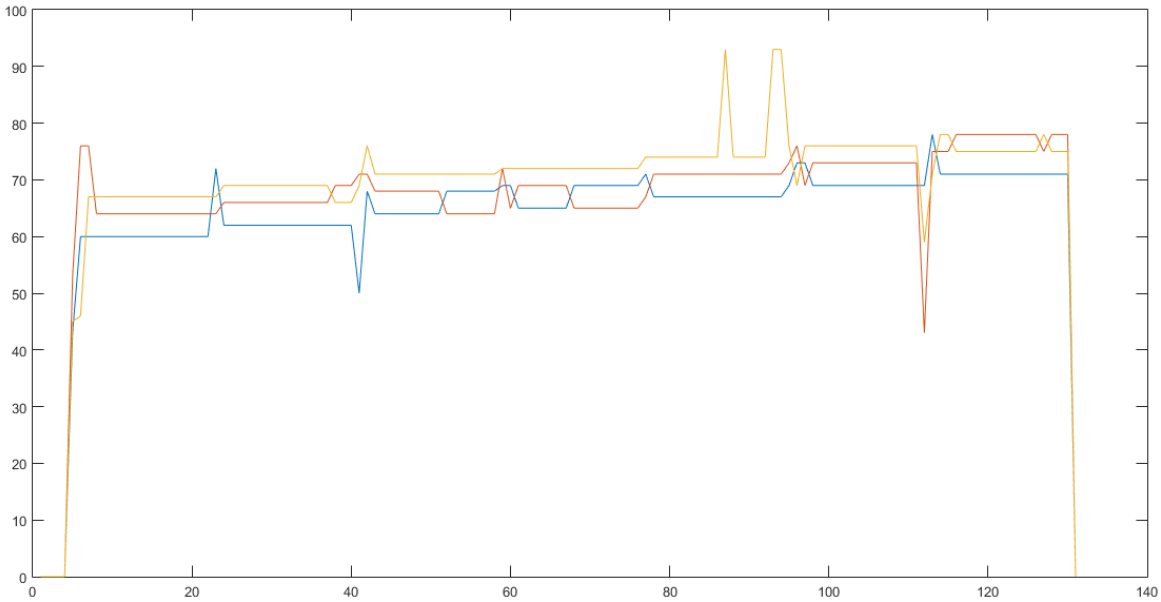


Figure 13: Output of the Multi F0 Estimation block

4.2. Block 2: Voice tracking

I have named this second block 'voice tracking' in reference to the concept of voices and movement in music theory, which served me as inspiration for this block's design.

Just to give a little bit of insight to music theory, we have what we call 'melodies' and 'harmonies'. A melody is one voice, a concatenation of sounds where one note leads to the next one, forming a path of notes (and silences):

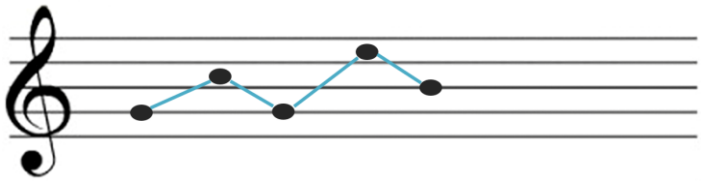


Figure 14: Melody

A harmony is formed by two or more voices sounding at once, and when we are playing an instrument we usually look at it in the form of chords:

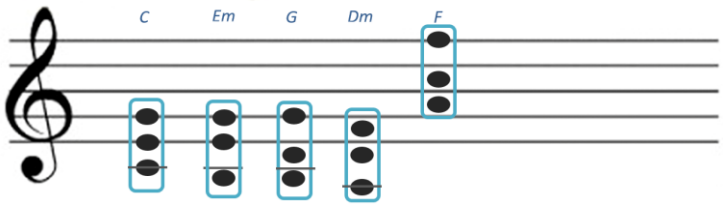


Figure 15: Harmony seen as a conjunction of chords

But with this definition, we can now see a harmony as a conjunction of independent voices or paths of notes and silences:

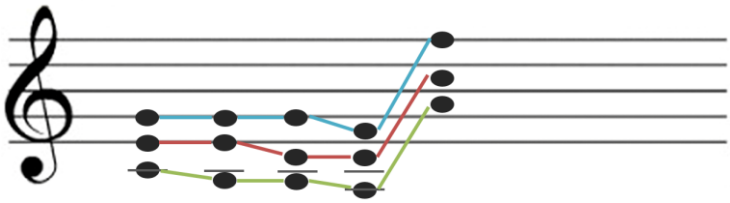


Figure 16: Harmony seen as a conjunction of independent voices

The task of the voice tracking block, then, is to order the notes extracted from the first block, joining the notes that belong to the same voice like in the picture above.

Figure 17 shows the diagram of the voice tracking block.

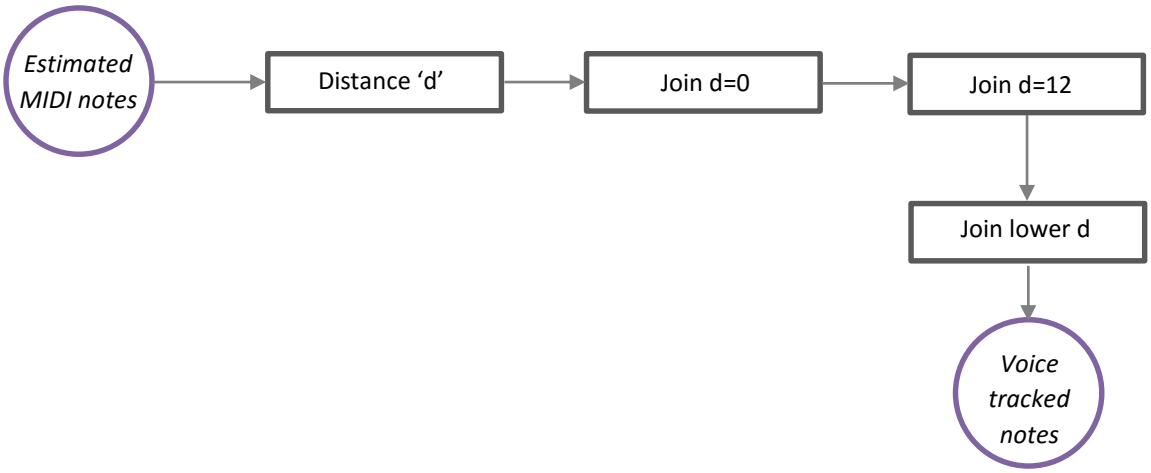


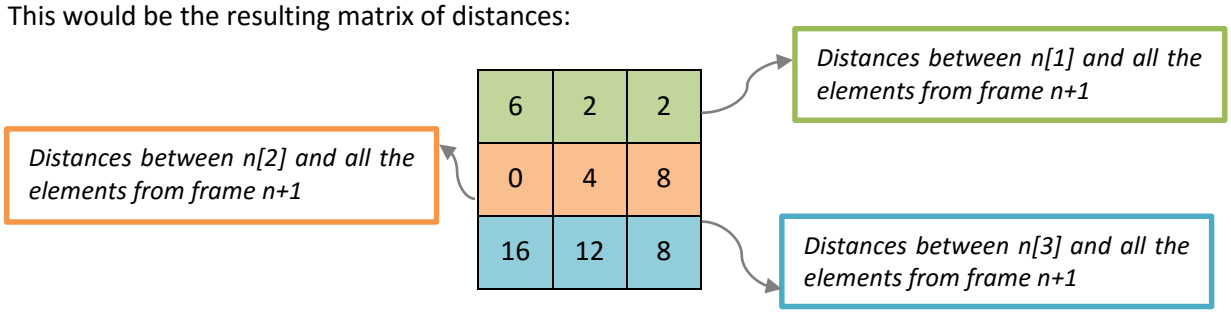
Figure 17: Voice tracking block diagram

1. First of all we compute the distance 'd' in semitones (absolute value) of every note from frame 'n' with every note from frame 'n+1' and create a matrix of distances between these 2 frames.

   For example, if we have the following frames 'n' and 'n+1':

n	n+1
66	60
60	64
76	68

This would be the resulting matrix of distances:



2. Then we look for those couples of notes with distance d=0 and join them (assign them to the same voice). If they have distance d=0 it means that they are actually the same note. Once that the join has been made we assign a very high value to this distance so it doesn't interfere with the next steps.
3. The second case we look for is distance d=12, which is the equivalent to an octave. By joining notes with d=12 we prevent errors of estimating a harmonic instead of the real fundamental frequency. This is a very common mistake in the F0 estimation. Again, once that the join has been made we assign a very high value to this distance so it doesn't interfere with the next step.
4. Now that we have joined d=0 and d=12, the remaining notes are joined to its lower distance partner (from low to high).
5. We repeat the process for every frame.

Following the previous example, the notes would be assigned like this:

n	n+1
60	60
66	64
76	68

Finally we end up with all the notes that belong to the same voice together in the same array.

The output of the voice tracking block is shown in figure 18. We can see the three voices with clear spurious mistakes, which will be corrected in block 4 by processing the onsets.

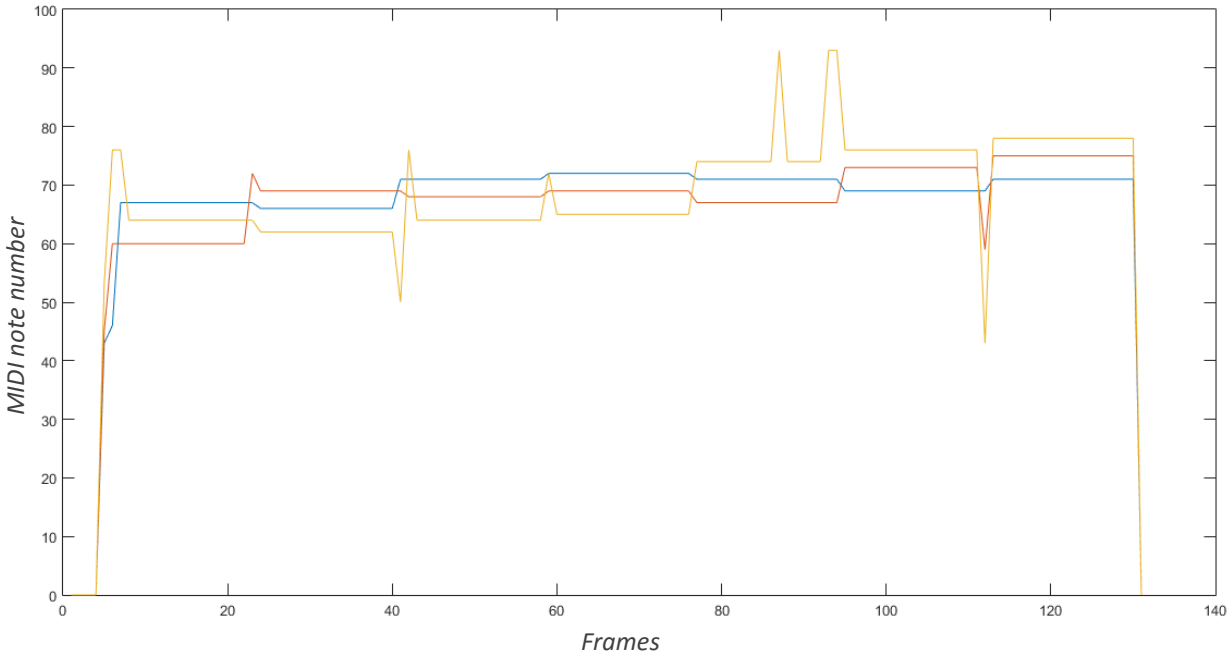


Figure 18: Output of the voice tracking block

4.3. Block 3: Onset detection

This third block processes the input audio signal and finds the frames in which an onset occur. It is based on the paper published by Matija Marolt in 2002 [2].

Figure 19 shows the diagram of the onset detection block.

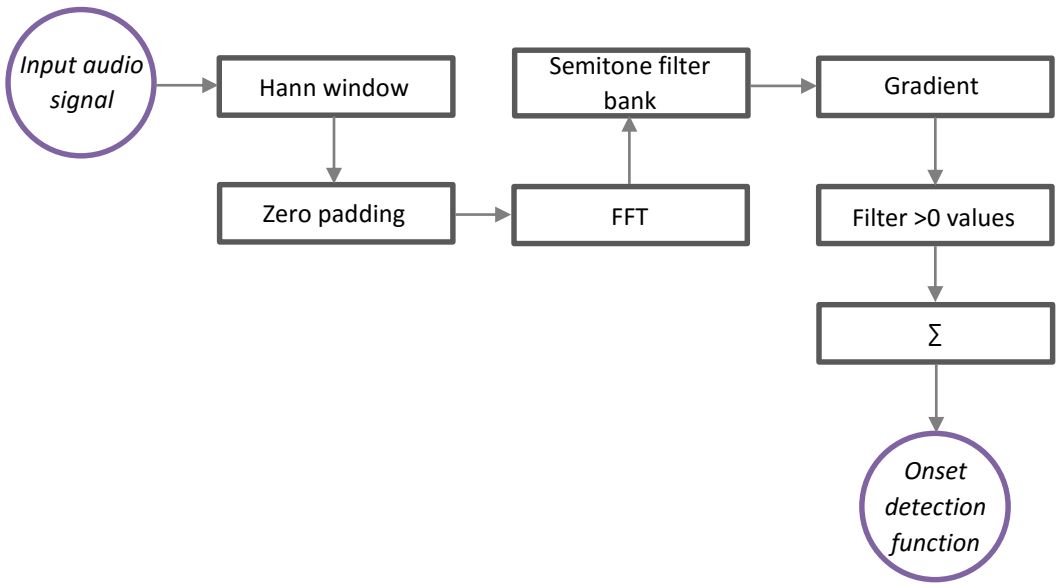


Figure 19: Onset detection block diagram

1. The first steps are very similar to those in the Multi F0 estimation block. First we apply a 92 ms Hann window with a 46 ms hop size to the input audio signal, then a zero padding to 4 times its length and finally a Fast Fourier Transform.

2. The next step is to pass the transformed signal through a filter bank, but this time it's a bank of triangular bandpass semitone filters whose center frequencies correspond to the fundamental frequencies of the musical notes.
3. For every frame *t* we obtain an array $b_i=\{b_1,b_2,\ldots,b_B\}$ where each bi is obtained by:

$$b_i = \sqrt{\sum_{k=0}^{K-1} (|X[k]| \cdot |H_i[k]|)^2}$$

4. Next we compute the gradient for each filter *i*:

$$c_i(t) = \frac{d}{dt} b_i(t)$$

5. We keep only the positive values from $c_i(t)$:

$$a(t) = \sum_{i=1}^{B} max\{0, c_i(t)\}$$

6. And finally we sum each $b_i(t)$ to obtain the onset detection function *o(t)*:

$$o(t) = \sum_{i=1}^{B} b_i(t)$$

Figure 20 shows the onset detection function *o(t)*. The peaks in the graphic show the frames where there is an onset. We can clearly appreciate the 7 onsets corresponding to the 7 chords.

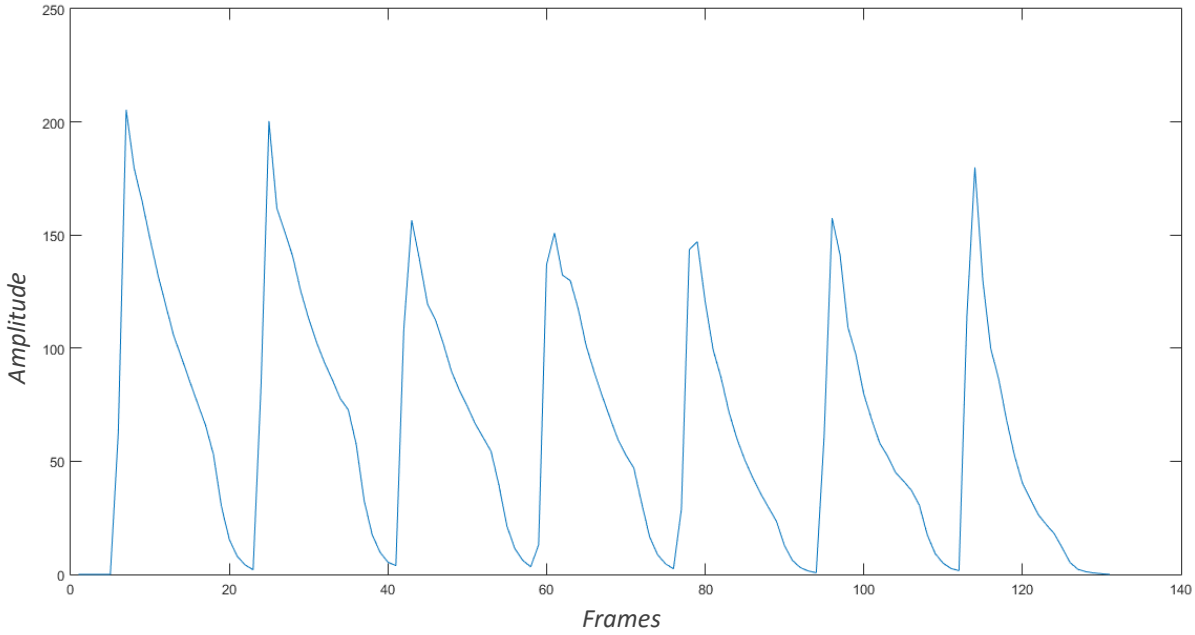


Figure 20: Output of the onset detection block

4.4. Block 4: Onset Processing

The onset processing block combines the outputs from blocks 2 and 3. The result is a cleaner and smoother output, eliminating possible spurious errors. Figure 21 shows this block's diagram.

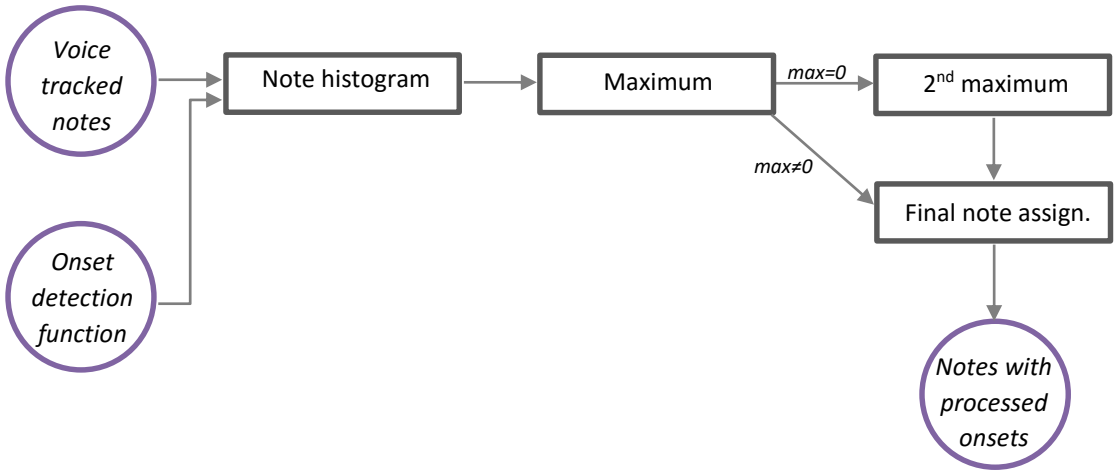


Figure 21: Onset processing block diagram

1. For each voice, we create an array with all the frames between an onset 'n' and onset 'n+1' and compute a note histogram. This histogram (figure 22) shows how many times every note appear in the array.

71	71	71	71	71	71	71	71	71	71	72	72	0	0	0	0	0	0

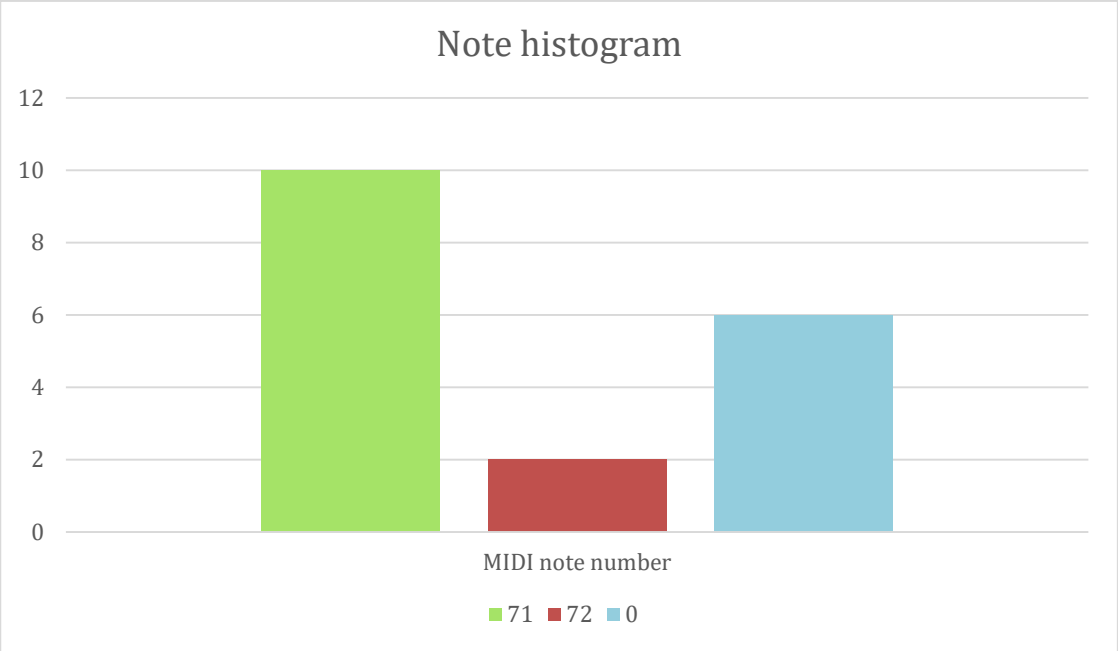


Figure 22: Note histogram

2. Once that we have the histogram, we look for the maximum value, which in this case is 10. If the maximum value is zero, we reject it and take the second maximum. The idea behind this method is to smooth the notes by removing spurious errors. We do that by replacing each note in the array with the note that has the maximum in the histogram (in this example is 71). This replacement is done note by note until a zero is found. From there, all remaining notes will be zeros. This is because a zero means there is no signal, and if there is no signal there won't be a new note until the next onset.
3. We repeat the process for every onset.

The resulting array in this example would be the following. The notes in blue are the ones that have been actually replaced.

71	71	71	71	71	71	71	71	71	71	71	71	0	0	0	0	0	0

Now that we have combined the voice tracking with the onset detection, we have the final notes, which are shown in figure 23. We observe a good note definition and tracking, without spurious errors between onsets.

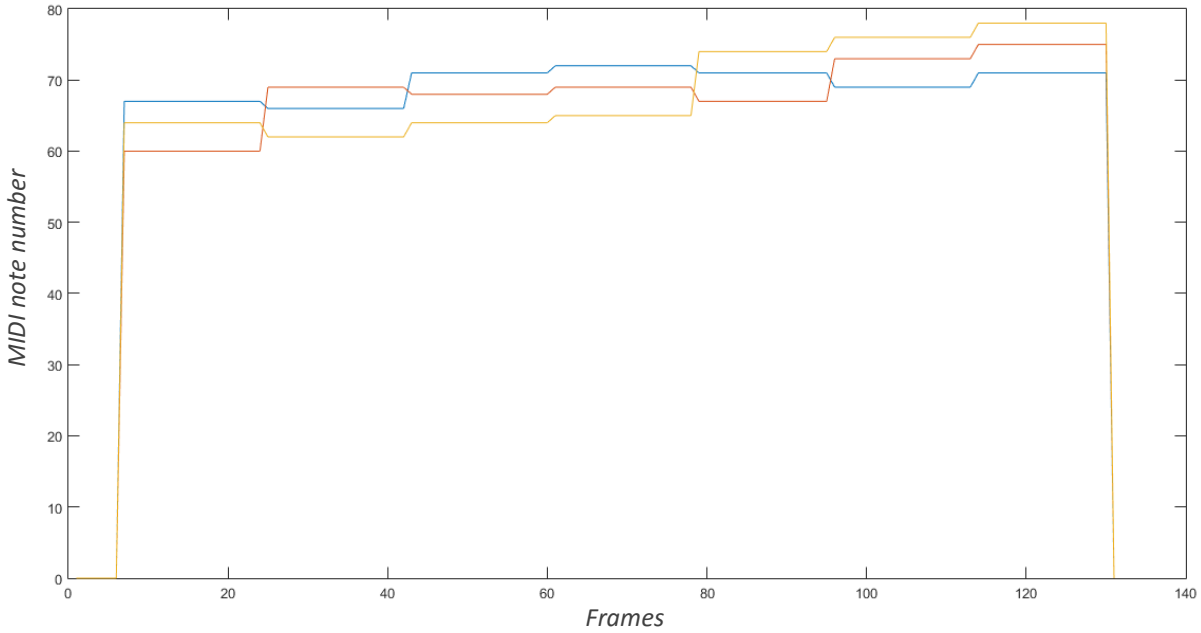


Figure 23: Output of the onset processing block

4.5. Block 5: Note tracking

The note tracking block is the final stage before the MIDI generation. It analyses the smoothed output from block 4 and creates a matrix containing the essential information of all individual notes: note number, note onset and note offset.

Figure 24 shows the diagram of the note tracking block.

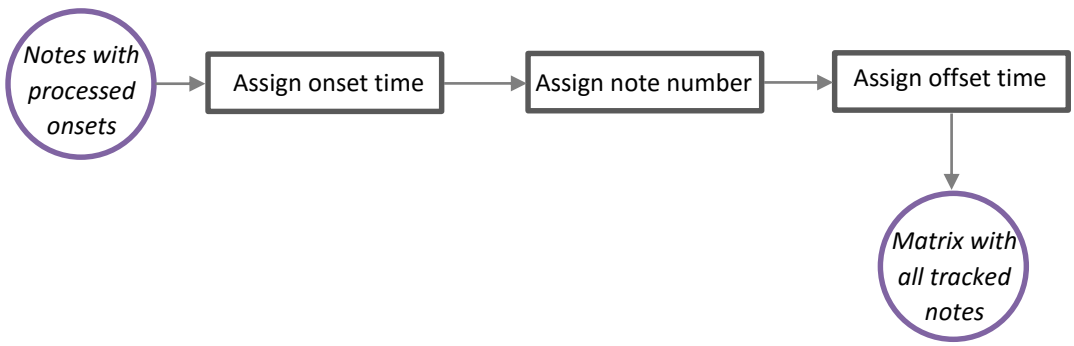


Figure 24: Note tracking block diagram

Using the information taken from the onset processing block we know where the onsets and offsets of the notes are. The only thing we have to do in this block is take all the information we already have and organize it in a matrix.

The figure below contains the output from this block. It has correctly detected the 21 notes corresponding to the 7 chords presented at the beginning.

MIDI note number	Note onset (seconds)	Note offset (seconds)
67	0,2760	1,0580
66	1,1040	1,8860
71	1,9320	2,7140
72	2,7600	3,5420
71	3,5880	4,3240
69	4,3700	5,1520
71	5,1980	5,9340
60	0,2760	1,0580
69	1,1040	1,8860
68	1,9320	2,7140
69	2,7600	3,5420
67	3,5880	4,3240
73	4,3700	5,1520
75	5,1980	5,9340
64	0,2760	1,0580
62	1,1040	1,8860
64	1,9320	2,7140
65	2,7600	3,5420
74	3,5880	4,3240
76	4,3700	5,1520
78	5,1980	5,9340

Table 7: Output of the note tracking block

4.6. Block 6: MIDI generation

This last block is the one that takes all the information on the notes and generates a MIDI file with it. This is the only block that I did not develop myself, it was developed by Ken Schutte in 2009 [19].

Figure 25 shows the diagram of the MIDI generation block.

Figure 25: MIDI generation block diagram

It takes as an input the matrix containing all the tracked notes and creates a MATLAB MIDI structure with all the information. Then it writes this MIDI structure into a MIDI file, which is the final output of the system.

In figure 26 we can finally see the representation of the MIDI file generated. In there we see the 7 seven chords over the time. Each green bar represents one note.

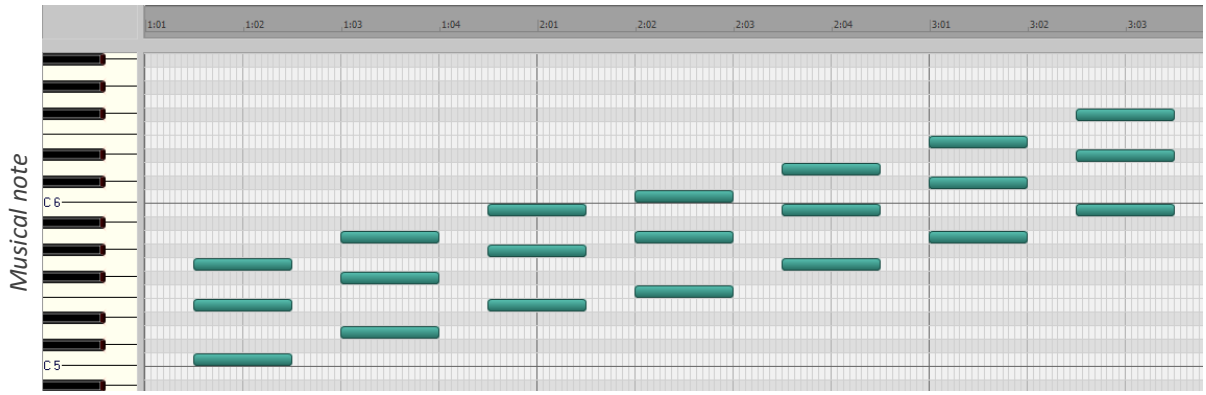


Figure 26: MIDI file generated. Output of the whole system.

5. Evaluation of the Transcription System

Throughout all the development process there has been an ongoing evaluation, which has been needed to optimize the algorithms and its parameters. In this section we are only going to present the final results.

In order to test the algorithms is necessary to set up an evaluation system. The final evaluation has been done with the MAPS database [18]. Each evaluation case has been tested with 125 recordings. By evaluation case we understand the following 10 cases, with polyphony ranging from 1 to 5 concurrent notes and with this information known or unknown by the transcription system:

1. Polyphony = 1 (fixed)
2. Polyphony = 2 (fixed)
3. Polyphony = 3 (fixed)
4. Polyphony = 4 (fixed)
5. Polyphony = 5 (fixed)
6. Polyphony = 1 (estimated)
7. Polyphony = 2 (estimated)
8. Polyphony = 3 (estimated)
9. Polyphony = 4 (estimated)
10. Polyphony = 5 (estimated)

The algorithm has been tested evaluating 3 parameters: Precision, recall and accuracy. These are common parameters for the evaluation of information retrieval algorithms and are defined as:

Precision: Answers the question "How many selected items are relevant?" Which translated to our case of study means "How many of the estimated notes are found in the ground truth?"

Precision is computed like this:

$$precision = \frac{TP}{TP + FP}$$

Where

$TP \stackrel{\text{def}}{=}$ *True Positive = Ground truth is not zero and estimated note is equal to the ground truth.*

$FP \stackrel{\text{def}}{=}$ *False Positive = An estimated note is not in the ground truth.*

Recall: Answers the question "How many relevant items are selected?" Which translated to our case of study means "How many items from the ground truth have been detected?"

Recall is computed like this:

$$recall = \frac{TP}{TP + FN}$$

Where

$TP \stackrel{\text{def}}{=}$ *True Positive = Ground truth is not zero and estimated note is equal to the ground truth.*

$FN \stackrel{\text{def}}{=}$ *False Negative = Ground truth is not zero and estimated note is zero.*

Accuracy: Is a measure that combines precision and recall in order to give a value that represents the global quality of the algorithm.

Accuracy is computed like this:

$$accuracy = \frac{TP}{TP + FN + FP}$$

Where

$TP \overset{\text{def}}{=}$ *True Positive = Ground truth is not zero and estimated note is equal to the ground truth.*

$FP \overset{\text{def}}{=}$ *False Positive = An estimated note is not in the ground truth.*

$FN \overset{\text{def}}{=}$ *False Negative = Ground truth is not zero and estimated note is zero.*

The two tables below contain the results obtained for the 10 cases mentioned above.

Fixing the polyphony (number of concurrent voices)

	Polyphony=1	*Polyphony=2*	*Polyphony=3*	*Polyphony=4*	*Polyphony=5*
Precision	0.8873	0.8350	0.8664	0.8818	0.7880
Recall	0.7421	0.6627	0.4950	0.3992	0.2730
Accuracy	0.7402	0.6142	0.4732	0.3868	0.2637

Table 8: Results fixing the polyphony

Estimating the polyphony

	Polyphony=1	*Polyphony=2*	*Polyphony=3*	*Polyphony=4*	*Polyphony=5*
Precision	0.8873	0.7692	0.6176	0.6639	0.6852
Recall	0.7421	0.4889	0.3661	0.2999	0.2714
Accuracy	0.7402	0.4375	0.3135	0.2694	0.2432

Table 9: Results estimating the polyphony

From these results we can extract several conclusions:

- The polyphony estimation is a critical step in the system since the accuracy is considerably lower when the system has to estimate the number of notes in the frame. However, for melodies (polyphony=1), the result does not vary. This means that there is much less ambiguity with the number of notes on monophonic recordings.
- The precision is always higher than the recall. This indicates the possibility of duplicates being detected.
- The accuracy of the algorithm decreases as the polyphony increases, which is not surprising. Higher polyphony means more notes to detect, which makes it more complex to analyse.

6. Budget

This part contains an estimation of the project's budget. It will take into account the costs of the personnel, hardware and software used.

The salary will be computed with the number of hours used by the student and the estimated cost of a junior engineer.

The software used has been MATLAB, so this budget will include the cost of the standard individual license, since it's the one that a company should purchase. For the hardware part, an average laptop computer with a 2nd generation Intel Core i5-2410M processor, 8GB of RAM and a dedicated NVidia GeForce GT 520MX graphics card.

Software	***Quantity***	***Price/Unit***	***Cost***
MATLAB standard individual license	1	2000 €	2000 €
Total			**2000 €**

Hardware	***Quantity***	***Price/Unit***	***Cost***
Laptop	1	600 €	600 €
Total			**600 €**

Personnel	***Price/hour (gross)***	***Hours/month***	***Salary/month***	***Months***	***Cost***
Junior engineer	8 €	100	800 €	8	6400 €
Supervisor	20 €	4	80 €	8	640 €
Total					**7040 €**

TOTAL	**9640 €**

Table 10: Budget

7. Conclusions and future development

Automatic Music Transcription is a task that can be approached in many different ways. Before starting to design and develop the algorithms, I have done extensive research on MIDI generation, VST development and techniques on the different parts that form the system, such as multiple F0 estimation, onset detection and note tracking.

In this thesis, I have designed and implemented a first prototype for a complete automatic music transcription system, combining techniques and methods explained in different papers with some new methods proposed in this work. Although it is not a final product, it provides an integrated solution that takes an audio signal as the input and generates a MIDI file as the output, which was the main goal, and it already gives some good results in simple scenarios such as melodies and low polyphonic recordings.

It would be interesting to consider future development because it is an idea that can have great success in the market if done properly, since it is a very handy tool for musicians. It is useful for a low budget approach to recording because it does not need of high quality microphones or room acoustics, since the conversion to MIDI makes it robust against noise. It is also useful for musicians who do not have a MIDI keyboard, because it works with any instrument, or even people who do cannot play any instrument, because it works with singing voice too. It could also bring a whole new world to music performances if the algorithm could be designed to work in real time with close to zero latency. In addition, there is room for many new features that can easily differentiate it from the competition.

From the evaluation we have seen that the system's bottleneck is the polyphony estimation, so in future development that would be the first part to improve. If we could get the algorithm to work as well in both cases (estimating and fixing the polyphony), that would already be a huge improvement in performance. The next step would probably be to improve the onset processing block so it affects only to the new notes instead of all the notes in the frame. And finally, it would be interesting to rethink the Multi F0 Estimation algorithm in order to make it more efficient and accurate. It could be interesting to combine the current idea with some probabilistic models that could train the system by applying harmony concepts.

Bibliography

[1] A. Klapuri. "Multiple Fundamental Frequency Estimation by Summing Harmonic Amplitudes". ISMIR, in proceedings of the 7th International Conference on Music Information Retrieval, Victoria, Canada, 8-12 October 2006: 216-221.

[2] M. Marolt, K. Alenka, M. Privosnik, and S. Divjak. "On Detecting Note Onsets in Piano Music." I*EEE MELECON, May 7-9, 2002, Cairo, EGYPT*: 385-89.

[3] A. Pertusa Ibáñez. "Computationally efficient methods for polyphonic music transcription". Ph.D. thesis, Department of Informatic Languages and Systems, Universidad de Alicante, Alicante, Spain, 2010.

[4] M. Bay, A.F. Ehmann, J.S. Downie. "Evaluation of Multiple-F0 Estimation and Tracking Systems". 10th International Society for Music Information Retrieval Conference (ISMIR 2009): 315-320.

[5] W. Lao, E.T. Tan, A.H. Kam. "Computationally Inexpensive and Effective Scheme for Automatic Transcription of Polyphonic Music". IEEE International Conference on Multimedia and Expo (ICME 2004): 1775-1777.

[6] H. Ming, D. Huang, L. Xie, H. Li. "Learning Optimal Features for Music Transcription". IEEE ChinaSIP 2014: 105-109.

[7] C.G.v.d. Boogaart, R. Lienhart. "Note Onset Detection for the Transcription of Polyphonic Piano Music". IEEE International Conference on Multimedia and Expo (ICME 2009): 446-449.

[8] J. Yin, T. Sim, Y. Wang, A. Shenoy. "Music Transcription Using an Instrument Model". ICASSP 2005, 217-220.

[9] M.P. Ryynänen, A. Klapuri. "Polyphonic Music Transcription Using Note Event Modeling". IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. October 16-19, 2005, New Paltz, NY.

[10] A. Klapuri, M. Davy. *Signal Processing Methods for Music Transcription*. New York: Springer, 2006.

[11] Justin J. Salamon. "Melody Extraction from Polyphonic Music Signals". Ph.D. thesis, Department of Information and Communication Technologies, Universitat Pompeu Fabra, Barcelona, Spain, 2013.

[12] Celemony. *Celemony (Melodyne)*. Web. <http://www.celemony.com/es/melodyne/what-is-melodyne>.

[13] Ableton. *Ableton Live*. Web. <https://www.ableton.com/en/live/>.

[14] Widisoft. *WIDI Recognition System*. Web. <http://widisoft.com/english/mp3-midi-products.html>.

[15] Martin Finke. "Making Audio Plugins." *Martin Finke's Blog. Music & Programming*. Web. <http://martin-finke.de/blog/tags/making_audio_plugins.html>.

[16] Steinberg."3rd Party Developer." *Steinberg SDKs*.Web. <http://www.steinberg.net/en/company/developers.html>.

[17] JUCE. "JUCE Tutorials." *JUCE*. Web. <https://www.juce.com/tutorials>.

[18] Telecom ParisTech. "MAPS Database – A Piano Database for Multipitch Estimation and Automatic Transcription of Music." 8 July 2010. Web. <http://www.tsi.telecom-paristech.fr/aao/en/2010/07/08/maps-database-a-piano-database-for-multipitch-estimation-and-automatic-transcription-of-music/>.

[19] Ken Schutte. "MATLAB and MIDI." *KenSchutte.com*. Web. <http://kenschutte.com/midi>.

[20] Michigan Technology University, Department Of Physics. "Frequencies for Equal-tempered Scale, A4 = 440 Hz." *Physics of Music - Notes*. Web. <http://www.phy.mtu.edu/~suits/notefreqs.html>.

[21] MIREX. "MIREX HOME." - *MIREX Wiki*. Web. 21 Apr. 2016. <http://www.music-ir.org/mirex/wiki/MIREX_HOME>.

Glossary

DAW	Digital Audio Workstation
F0	Fundamental Frequency
FFT	Fast Fourier Transform
FN	False Negative
FP	False Positive
MAPS	MIDI Aligned Piano Sounds
MIDI	Musical Instrument Digital Interface
MIREX	Music Information Retrieval Evaluation eXchange
Monophony	Only one note sounding at once
Polyphony	Two or more notes sounding at once
Note onset	Where a note begins
Note offset	Where a note ends
TP	True Positive
VST	Virtual Studio Technology