FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Interactive Manipulation of Musical Melody in Audio Recordings

Miguel Miranda Guedes da Rocha e Silva

Master in Electrical and Computers Engineering

Supervisor in FEUP: Rui Penha (PhD)

Supervisor in INESC: Matthew Davies (PhD)

June 25, 2017

Resumo

Com a remistura a tornar-se uma forma mais prevalente de expressão musical e *software* de edição de música cada vez mais disponível para uma maior audiência, mais pessoas estão envolvidas na produção e no compartilhamento de suas próprias criações. Assim, este projeto foi desenvolvido com os utilizadores finais em mente, incluindo aqueles que não possuem o conhecimento técnico de processamento de áudio, mas o interesse em fazer experiências com as suas coleções musicais.

O objetivo deste projeto é desenvolver uma técnica interativa para manipular melodia em gravações musicais. A metodologia proposta baseia-se no uso de métodos de detecção de melodia combinada com a utilização da *invertible constant Q transform* (CQT), que permite uma modificação de alta qualidade do conteúdo musical. Este trabalho consiste em várias partes, a primeira das quais concentra-se na extração da melodia de gravações polifónicas de áudio e, posteriormente exploramos métodos para manipular essas gravações. O objetivo a longo prazo é alterar uma melodia de um pedaço de música de forma a que pareça semelhante a outra. Definimos, como objetivo final para este projeto, permitir aos utilizadores realizar a manipulação de melodia e experimentar com sua coleção de músicas. Para atingir esse objetivo, desenvolvemos abordagens para manipulação de melodia polifónica de alta qualidade, usando conteúdo melódico e gravações de áudio misturadas. Para avaliar a usabilidade do sistema, foi realizado um estudo de utilizador.

Abstract

With remixing becoming a more prevalent form of musical expression and music editing software becoming more readily available for a larger audience, more people are engaging in the production and sharing of their own creations. Thus, this project was developed with the end users in mind, including those who may not possess the technical ability in audio processing but may want to experiment with their music collections.

The objective of this project is to develop an interactive technique to manipulate melody in musical recordings. The proposed methodology is based on the use of melody detection methods combined with the invertible constant Q transform (CQT), which allows a high-quality modification of musical content. This work consists of several stages, the first of which focuses on extracting the melody from polyphonic audio recordings and, subsequently we explore methods to manipulate those recordings. The long-term objective is to alter a melody of a piece of music in such a way that it may sound similar to another. We set, as an end goal for this project, to allow users to perform melody manipulation and experiment with their music collection. To achieve this goal, we devised approaches for high quality polyphonic melody manipulation, using melodic content and mixed audio recordings. To evaluate the system's usability, user-study evaluation of the algorithm was performed.

Agradecimentos

I would like to thank Prof. Rui Penha and INESC-TEC Porto for this opportunity and Prof. Matthew Davies for the much needed help and guidance to me provided during this year. I would also like to thank the twelve people who offered their time to partake in the experiment for this project.

Miguel Rocha e Silva

Contents

List of Figures

List of Acronyms

CQT	Constant Q Transform
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
HMM	Hidden Markov Model
HPSS	Harmonic/Percussive Sound Separation
MIDI	Musical Instrument Digital Interface
MIR	Music Information Retrieval
NSGT	Non-Stationary Gabor Transform

Chapter 1

Introduction

1.1 Motivation

Melody is one the most defining characteristics of a piece of music, particularly those in popular music and jazz genres. In this project, we propose an algorithm to take the melody of a musical recording and change its components, modifying their pitch or deleting them altogether and then re-synthesizing and playing back the results. Two novel properties of our system are: i) the ability to identify and then shift all the instances of a given note at the same time, and ii) to suggest a set of possible pitch transformations for a given note. We seek to accomplish not only the manipulation and transformation of melody, but also the development of an application that allows users to experiment with existing musical content regardless of whether they have the technical background to understand the inner workings of the system. In other words, our work aims to provide a straightforward means for music enthusiasts to be creatively engaged within musical content.

1.2 Context and Overview

One of the critical aspects of this work is the extraction of melody from a musical recording to manipulate it by applying a pitch shift to given notes of the main melody. To achieve this, several methods that extract pitch, and subsequently melody, have been proposed. Pitch shifting is the process of altering a recorded melody so that its notes are changed, while the rest of the music remains the same. This technique is commonly used for pitch correction of musical performances or for transposing a piece of music to a different key [1]. Pitch shifting is highly relevant to this work and it has been shown recently that it can be performed using the well-known constant Q transform (CQT) [2]. While pitch shifting is widely used to apply a global change to a piece of music (i.e. to shift all pitched and unpitched content at the same time), a particular novelty of our approach is the ability to selectively apply pitch shifting only on isolated notes of the melody – while keeping everything else constant.

In our work, we begin by estimating the melody using the state of the art MELODIA Vamp plugin [3] for Sonic Visualiser. Here the input's main melody is automatically annotated through the identification of its fundamental frequency, and a melody activation function calculated using the salience based approach for melody detection. Next, we use the MATLAB toolbox developed by [1] which includes an implementation of the invertible constant Q transform [4] and contains a pitch shifting tool – normally used for global pitch shifts (i.e. to all notes at once). We integrate the extracted melody contour into the CQT representation and then pursue its manipulation via pitch shifting of individual notes in the CQT representation. This is achieved by quantizing the melody contour into the CQT, with a chosen number of bins. Next, the bins that contain melody and its harmonics that are automatically identified, to more accurately preserve its original characteristics when transforming it. Prior to pitch shifting, a binary mask is generated which isolates the main melody and its harmonics. The transformations to the melody are then done by selecting notes from the melody (along with their harmonics) and altering their pitch via the pitch shifting functionality in the CQT toolbox. Via an interactive interface, we allow users to specify which notes they wish to modify, as well as giving them the option to capture all instances of the same note, and providing suggestions as to which pitch shifts to apply to any user selected note.

To evaluate the effectiveness of the system we ran an experiment where users performed a set of simple melody manipulation operations using the interface and then they answered a survey with questions about their user experience, as well as the quality of the result. The answers to the survey indicate that the system was successful in producing high quality results of manipulated melody, while also displaying a wide variety of opportunities to improve and explore means to creatively modify musical content.

1.3 Structure of the Document

In chapter 2, we describe the state of the art and we present research related to this topic, as well as existing systems for manipulating melody. Chapter 3 describes the development of the melody manipulation system. Chapter 4 details how the evaluation of the the system was approached and what results were obtained. In chapter 5, we provide some final remarks and discuss future work.

1.4 Publication Resulting from this Dissertation

Miguel Rocha e Silva, Matthew Davies, Rui Penha. "Interactive Manipulation of Musical Melody in Audio Recordings," *In 2nd Doctoral Congress in Engineering (DCE)*, 2017.

Chapter 2

State of the Art

In this chapter we present the research which served as basis for the development of this project. Here we cover approaches for pitch and melody extraction, including a few methods for the latter. We also cover the theory behind the constant Q transform, which we use as a core signal processing tool. Finally we mention existing systems that apply some of same principles as the one we proposing.

2.1 Music Information Retrieval

Music Information Retrieval (MIR) can be described as "a multidisciplinary research field that draws upon the traditions, methodologies, and techniques of a remarkably wide range of disciplines" [5]. One of its early goals was to provide an equal or higher level of access to the vast range of music than was available through text-based search engines [6]. MIR research seeks to develop retrieval systems with which users interact using the music itself, presented in auditory (MP3, WAV, etc.) and/or symbolic (MIDI, score, etc.) formats, allowing them to make musical queries. A few examples of musically framed queries include: query-by-singing, where the user sings or hums part of the piece of music they are looking for, query-by-example, where the user submits a known MP3 to find similar pieces and query-by-notation, where the user places notes on a musical staff to form the query.

Another aim of MIR is to devise a way to automatically identify the genre and mood of a song and also if it is a cover version [7]. This relies on using techniques such as chord detection (and key recognition), structural segmentation and annotation of songs, and analysing musical similarity. This last one also is one of the basis for music recommendation systems.

Computerized music transcription is used to generate music notation and can be used to assist in composition [8], while beat tracking and beat extraction are done to make automatic drum beats [9]. Of particular interest for this project are the techniques to detect and extract melodic content.

2.2 Melody and Pitch Extraction

One of the critical aspects of this project is the extraction of melody from a polyphonic musical recording in order to, then, manipulate it. To achieve this, several methods that extract pitch, and subsequently melody, have been proposed.

2.2.1 Pitch Estimation

The precursor to melody extraction is monophonic pitch estimation (also referred to as pitch tracking), which is strongly based on fundamental frequency f_0 estimation. Pitch tracking algorithms can be classified as either time-domain or frequency-domain algorithm, per the domain in which they are processed. Time-domain algorithms attempt to extract the pitch directly from the signal waveform by estimating its periodicity. One method is to apply an autocorrelation function to the signal [10], [11]. For periodic signals, the maxima of the autocorrelation function correspond to the period of the f_0. Some earlier methods [12], [13] attempted to translate human auditory system models to extract the perceived pitch of a signal. Frequency-domain algorithms transform the signal, usually using a short-time Fourier Transform, into its spectral representation to estimate its f_0. One of the earliest examples of such algorithms was based on analysis of the inverse Fourier transform of the logarithm of the power spectrum of the signal, called the cepstrum [14]. Here, a strong peak of a periodic signal is in the location corresponding to the lag of the period, which can be used to compute the signal's f_0. Other methods exist where the magnitude spectrum peaks are matched against where they are expected to be in the harmonics of an f_0. This is referred to as harmonic matching [15], [16].

2.2.2 Salience Based Approaches

The most widely used method for melody detection is the salience based method. This approach relies on the creation of a function which can identify pitch candidates in polyphonic signals. Although there are several different published variations on this approach, they consist of the same basic stages.

In an initial stage, most approaches attempt to highlight the frequencies most likely to contain melodic content by using a filter. In the following phase the spectral representation of several frames of the signal is obtained. This can be done by applying a short-time Fourier Transform with a window large enough (50-100 ms) to provide the necessary frequency resolution to tell different notes apart and necessary time resolution to track quick changes in pitch. However, by using the constant-Q transform [2], [17] or other multiresolution transforms, such as a multirate filter bank [18] or the multi-resolution FFT [19], [20], [21] these resolution limitations can be overcome. After the transform, some methods do some additional processing based on spectral peaks, filtering out the ones that do not represent melody [18], [20], [22], [21]. The salience function, which outputs the possible candidates for the melody, can be obtained in various ways, the most common of which being the summation of the amplitude and its harmonic frequencies [17], [20], [23], [3],

[21]. In the final phases, the peaks of the salience function are processed to determine which ones belong to the melody. Some authors [17], [22], [3] propose grouping the continuous pitch contours. The final melody can be obtained using several different tracking techniques, such as clustering [22], Hidden Markov Models (HMM) [23], [21], dynamic programming [24], [20] or heuristic-based tracking agents [18], [25]. One last stage, called voicing detection, is used to determine when the main melody is being played and when it is not. Some approaches do this by using a fixed or dynamic per-frame salience-based threshold [22], [17], [25] and [24] , while [23] incorporate a silence model into the tracking part of the HMM algorithm.

2.2.3 Source Separation Based Approaches

Another strategy is to separate the melody from the remainder of the mix. One method, proposed by [26], models the power spectrogram of the signal as a sum of the lead voice and accompaniment contributions. The former is represented with a source/filter model, while the latter is represented as the sum of an arbitrary number of sources with distinct shapes. Two representations for the source/filter model have been proposed: an extended Instantaneous Mixture Model, which represents the lead melody as the mixture of all possible notes, and a Gaussian Scaled Mixture Model, which allows only one source/filter couple to be active at any given time. After the model estimation, the Viterbi algorithm is used to obtain the final melody frequency sequence. In this case voicing is done using Wiener filtering [27] to separate the estimated melody signal and computing the energy at every frame to establish a threshold for frame where there is melody. Another method, proposed by [28], uses the Harmonic/Percussive Sound Separation (HPSS) algorithm [29] to, firstly, separate sustained chords from the more rapidly changing musical content, and then to separate the percussive elements from the main melody. Given that the melody is enhanced after the algorithm is applied twice, its frequency sequence can be obtained using dynamic programming. Finally, voicing detection is done by setting a threshold on the distance between the melody signal and the percussive signal, which are produced the second time the HPSS algorithm is run.

2.2.4 Marsyas

Marsyas[1] is an open source software designed by George Tzanetakis and collaborators with emphasis on Music Information Retrieval. It is a framework for creating programs that process streams of number, particularly suited for streams that represent sound or audio characteristics such as melody, harmony, rhythm, etc. The system allows for different experiences according to whether the user is an expert or a novice. Through a graphic interface and without having to compile code, a novice user is able to use control to experiment with networks of primitive objects, while expert users can write code create new primitive objects and more complex application.

[1]http://marsyas.info/

2.2.5 Contour Based Approaches

This algorithm is related to the salience based approaches described above and it is centred on the creation and characterization of pitch contours [30]. First the non-salient peaks are filtered out to minimize the existence of non-melody contours. The peaks are grouped in contours, starting by selecting the highest peak and adding to a new pitch contour and then tracking forward in time to search for the next peak. This process is repeated until there are no more matching salient peaks to be found. After the contours are created and characterized accordingly, the next step is to filter out the non-melody contours by performing voicing detection similar to what was described above and octave error correction, which removes pitches commonly referred to as "ghost pitches" that have an f_0 that is a multiple of the actual pitch, which can cause the algorithm to select a pitch that is either one octave above or below the actual melody. The final melody selection is done from the remaining contours of each frame.

2.2.6 MELODIA

Presented in 2012, MELODIA is an implementation of the work by [3] in the form of a Vamp plug-in for Sonic Visualiser[2]. This plug-in automatically estimates the fundamental frequency of a corresponding to the pitch of the main melody of a polyphonic musical recording, also performing voicing detection to identify when there is melody and when there is not. MELODIA receives as input an audio file in a compatible format, such as .mp3, .wav, .ogg, etc., and outputs the pitch of the main melody (or rather its correspondent frequency), a representation of the salience function and a representation of all pitch contours, some of which are identified as melody contours. Figure 2.1 shows the results of this process. As can be observed, the green lines correspond to the estimated f_0 of the melody at a given time and when there is no melody detected the estimated frequency output is zero. Because it is freely available and Sonic Visualiser has the ability to synthesize and play back the results, which it computes with high accuracy, MELODIA has become an attractive tool for this project.

2.3 Methods for Transforming Sound

After the melody is extracted, there are various ways it can be manipulated. These algorithms are based on MIR research and are applied to modify specific characteristics of musical content, such as pitch, rhythm and timbre.

Time stretching is a process through which a sound is made to be longer or shorter without affect to its pitch. It can be approached by either time-domain methods or frequency-domain methods, the latter being more appropriate for polyphonic signals and are mostly based on standard phase vocoder algorithms [31].

Conversely, pitch shifting refers to the changing of pitch of a sound, making it higher or lower, without affecting the tempo. Pitch shifting is commonly used for pitch correction of musical

[2]http://sonicvisualiser.org/

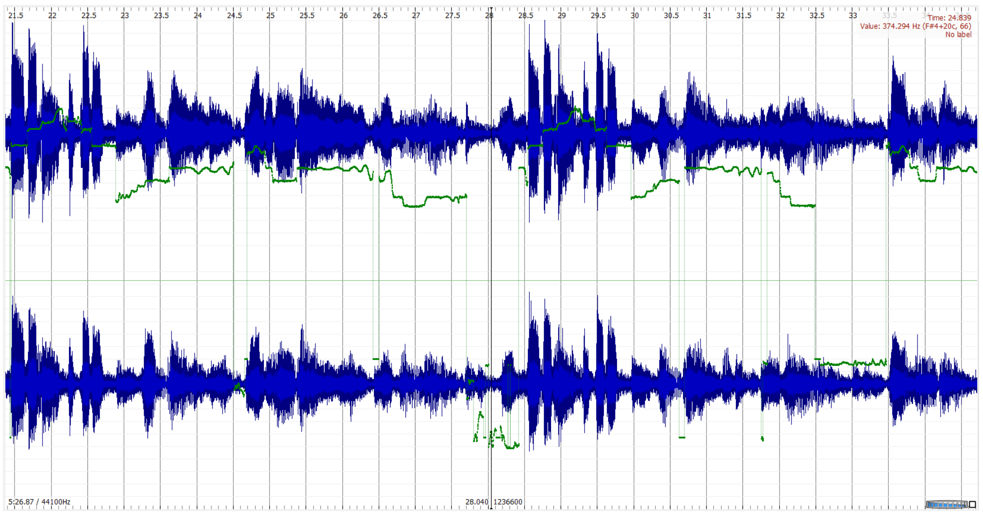


Figure 2.1: Display showing the sound wave of a file (blue) and its respective melody as detected by MELODIA (green)

performances or for transposing a piece of music to a different key [1]. A prime example of pitch shifting is Autotune[3], where recorded vocals are altered so that they become perfectly in tune with the rest of the music. Pitch shifting is highly relevant to this project and can be done by using the constant Q transform, which will be discussed in section 2.3.1.

Sound morphing is the timbral manipulation of a sound and its components, changing the sound's defining characteristics. In the traditional sense, it can be done by manipulation the partial frequencies of a sound and interpolation of the time-varying frequencies and amplitudes of corresponding partials of the original sound [32]. Cross-synthesis is an example of a sound morphing technique, where one parameter of a synthesis model is applied together with another parameter of another synthesis model. Examples of such models include linear predictive coding, physical modelling and the vocoder [33].

2.3.1 Constant Q Transform

Traditionally, signal processing of music signals is done using the Fourier Transform that, although computationally efficient, is not able to map musical frequencies in the most useful way, given its filter banks are separated by a constant frequency.

The constant Q transform (CQT), introduced by [2] and closely related to the Fourier transform, also uses a bank of filters. However, unlike the Fourier transform, these have geometrically spaced frequencies. This is especially useful in the context of signal processing of musical signals, as by choosing an adequate number of filter per octave, we can obtain a representation that is similar to the equal tempered scale commonly used in Western Music. The constant Q transform has finer time resolution at higher frequencies, mimicking the way sound is perceived by the human auditory system.

[3]http://www.antarestech.com/

A CQT of an audio input $x[n]$ can be calculated directly by evaluating

$$X[n] = \frac{1}{N[k]} \sum_{n=0}^{N[n]-1} W[k,n]\, x[n]\, e^{-\frac{j2\pi Qn}{N[k]}}, \tag{2.1}$$

where the quality factor Q is given by

$$Q = f_k/(\Delta f_k), \tag{2.2}$$

with a bandwidth Δf_k and a frequency f_k given by

$$f_k = 2^{k/B} f_{min}, \tag{2.3}$$

where $k = 0, 1, \ldots$ is the spectral component and B is the number of filters per octave and f_{min} is the minimum frequency. The window size $N[k]$ is calculated by

$$N[k] = \frac{S}{\Delta f_k} = \frac{S}{f_k} Q, \tag{2.4}$$

where $S = 1/T$ and T is the sample time. Equation 2.1 is obtained using 2.3, 2.2 and 2.4 as constraints to a short time Fourier transform [34]

$$X[k] = \sum_{n=0}^{N-1} W[n]\, x[n]\, e^{-\frac{j2\pi kn}{N}}. \tag{2.5}$$

In the case of 2.1, the period in samples is $N[k]/Q$, analysing Q cycles every time. The period becomes $2\pi Q/N[k]$ and the window function $W[n]$, although having the same shape for each component, its length now depends on $N[k]$, therefore it is now also a function of k. Finally, as the number of terms varies with k, it is also necessary to normalize the sum, dividing it by $N[k]$.

2.3.2 Invertible Constant Q Transform

One of the major drawbacks of the CQT described above is that it lacked an inverse function, making it impossible to reconstruct the original signals once the transform was applied. An invertible approach to the CQT proposed by [4] is similar to the forward transform but done in reverse order. A reconstructed signal $y_d[n]$ is computed for each octave of the CQT

$$Y_d = V X_d, \tag{2.6}$$

where X_d is the d-th octave of the transformed signal, V is the inverse spectral kernel, which corresponds to the conjugate transpose of the kernel used in the efficient CQT algorithm [35]. The column m of Y_d is the DFT approximation of the same column of X_d for the octave's frequencies, otherwise Y_d is zero. The signal is added to another signal that contains the reconstruction of all the lower octaves. The resulting signal $\hat{x}_d[n]$ is upsampled by a factor of two, multiplied by two and then low-pass filtered with a cutoff frequency of $f_k/4$. After this process is repeated for every octave, we get a signal $\hat{x}[n]$ that is an approximate reconstruction of the original input $x[n]$.

Another method to compute the CQT and its the inverse function uses non-stationary Gabor transform (NSGT) [36]. This approach allows windows with adaptive bandwidth and time-shift parameters to be chosen according to the bandwidth of each window. The shifted versions of the basic window functions used in the transform are computed in the frequency domain, which allows an efficient implementation of the FFT.

2.3.3 Pitch Shifting Using the Constant Q Transform

There is a method, devised by [1] to perform pitch shifting using the CQT. Shifting the CQT by r CQT bins, the frequency of a spectral peak in the CQT representation is scaled by a factor α. The shift in CQT bins is given by

$$r = B\log_2(\alpha), \tag{2.7}$$

where B is the resolution of the CQT and r is independent from the frequency of the spectral peak. The viability of shifting the CQT coefficients depends on the placement of sampling point in both the time and frequency domains. The hop size H_k increases from one atom to the next for increasing frequency-bin indexes k, which results in a constant overlap factor between window functions in the time domain and window lengths N_k that decrease with increasing k. Although phase coherence is lost when the CQT coefficients are shifted along the frequency dimension, it can be compensated by multiplying all coefficients within the same region with

$$Z = e^{i\Delta f_{k,u} H_k}, \tag{2.8}$$

where $\Delta f_{k,u}$ is the difference between the old and new peak bin in column u of CQT. Finally, the output signal is reconstructed using the inverse CQT.

2.4 Existing Systems for Melody Manipulation

2.4.1 SPEAR

SPEAR[4] is an audio analysis, editing and synthesis application which deconstructs a sound and represents it as many individual sinusoidal waves. A sound nearly identical to the original sound can be obtained by adding together all these partials. SPEAR uses a variation of the McAulay-Quatieri technique to compute the representation and linear prediction to determine the continuation of the sinusoid signals. Its interface enables manipulation and editing of sounds in a very flexible manner, allowing the user to work on each of the partials independently. Furthermore, it also has the ability to listen to the transformations without a synthesis or processing stage beforehand [37].

[4] http://www.klingbeil.com/spear/

2.4.2 Melodyne

Developed by Celemony, Melodyne[5] is a software that allows users to manipulate musical recordings. Here the musical notes are displayed as “blobs” that the user can drag, stretch, compress, delete, etc. while keeping a natural sounding recording. Melodyne contains specific algorithms for the different types of audio, for instance vocals, instruments, percussion, unpitched sounds and entire mixes. Pitch is distinguished in centre pitch, pitch modulation (i.e. vibrato) and pitch drift. The notes can be shifted along a myriad of different scales, while the connection between them remains unspoiled. Notes can be made longer, shorter or even quantized per specific time grids that can also be adjusted to fit the performance tempo. Time stretching is done so that, in the case of vocal edition, only the vowel sounds are stretched leaving the consonants intact. Another of Melodyne’s features is the ability to access an alter the overtone structures of a track, allowing the manipulation of the timbre of voices and instruments with high precision.

2.5 Summary

Although, in terms of melody manipulation, Melodyne is the closest existing system to the proposed one, the novelty of this project lies on several key aspects. We propose an open system, with a well-documented development process and the technical aspects behind its functionality, as well as the system’s evaluation methods and their results.

[5] http://www.celemony.com/en/melodyne/what-is-melodyne

Chapter 3

Methodology

In this chapter we present a detailed explanation of the different stages we went through in the development of this project. We divided this chapter in two parts: in the first part, we detail the back-end side of the project, from the extraction of melody to the development of code in MATLAB; the second part details the user interface, also developed in MATLAB, and its functionality.

3.1 Back-End

This project was developed following several stages. As ilustrated in figure 3.1, in the first stage, we used the MELODIA Vamp plug-in to extract the melody from various musical excerpts. The next stage is the computation of the Constant-Q Transform (CQT) of the musical excerpt and process the information extracted from MELODIA. From this, the following stage is the separation of the melodic content from the remainder of the track. We are then able to manipulate the melody by changing its notes. Finally, the last stage is the reconstruction of the musical recording and computation of the inverse CQT. In this chapter, we present an in-depth description of these stages.

3.1.1 Extracting the melody

Firstly, musical excerpts were run through MELODIA to extract their melodic content. MELODIA outputs a representation of the pitch contours of the melody. To adjust the accuracy of the melody detection, MELODIA offers several controls to change the value of relevant parameters such as minimum and maximum frequency, voicing tolerance, and the option to apply the algorithm to either a monophonic or polyphonic recording, as seen in figure 3.2. These default settings are not always ideal, so to reach a point when the detected melody was concurrent with the original, we had to engage in a trial-and-error experiment. After the melody is detected with sufficient accuracy, the pitch values and the time of their occurrence are extracted into a .csv file. This file contains the value of the fundamental frequency of the detected melody and the respective time stamp when it occurs.

We verified that recordings that exhibit a clear main melody, usually performed by a soloist, and are instrumental pieces were better suited for the MELODIA algorithm. Because of this,

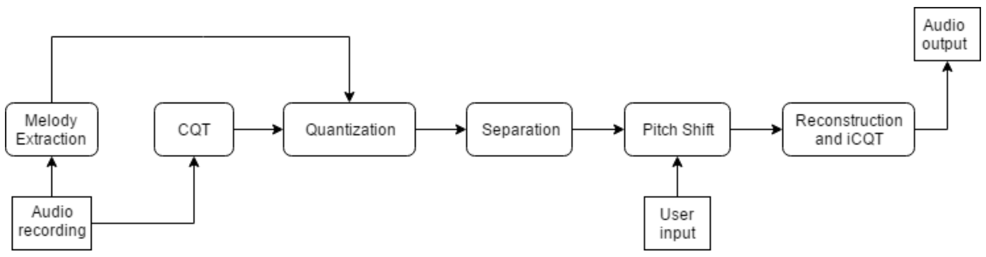


Figure 3.1: Flow diagram

certain genres of music, namely jazz sub-genres such as Cool Jazz and Modal Jazz yield better results when through the algorithm, rather than Pop or Rock music, or even vocal music in general.

3.1.2 Computing the Constant-Q Transform

To compute the CQT, we used the MATLAB toolbox developed by [4]. The minimum frequency was set as 27.5Hz and the maximum frequency is half the sampling frequency f_s of 44100Hz. The number of bins per octave B is 48, which corresponds to 4 bins per semitone. Figure 3.3 shows the CQT of a musical signal with the melodic values extracted onto the .csv file plotted on top of it. To effectively separate the bins where the melodic values are located, we perform a quantization of said values.

3.1.3 Quantization

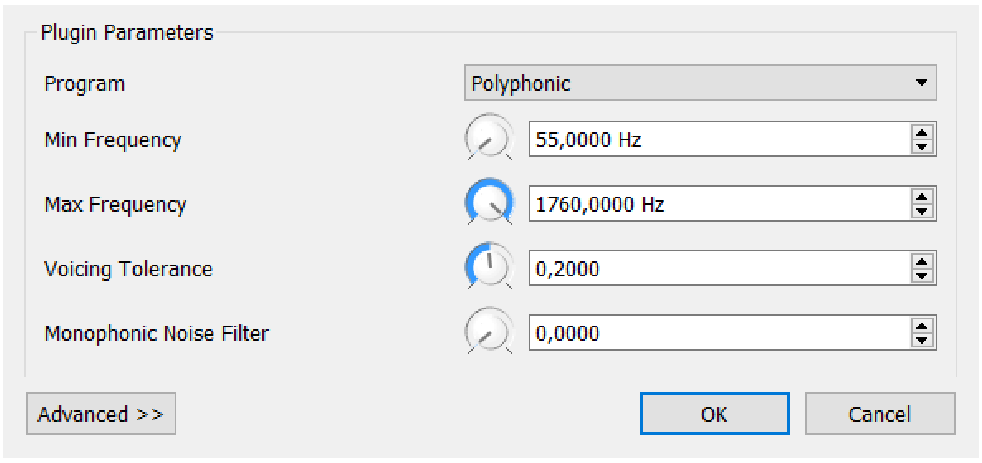
After the CQT is computed, the melodic values extracted from MELODIA are put into an array and the values corresponding to absence of melody are turned into zeroes. In this case, when MELODIA is unable to detect the presence of melody it assumes the frequency value to be -440Hz. To ensure the extracted melody has the same temporal resolution as the CQT, we interpolated the melodic values. Then, we calculated the frequency intervals which correspond to each of the CQT bins. This was done by applying the function

$$f_k = 2^{k/B} f_{min} \tag{3.1}$$

for every bin k until the maximum number of bins b_{max}, given by

$$b_{max} = \lfloor \log_2 \left(\frac{f_{max}}{f_{min}} \right) B + 1 \rfloor, \tag{3.2}$$

was reached. Once all these parameters were defined we proceeded to quantize the melody to the CQT bins. In this stage, for every melodic value we verify which bin frequency interval it belongs to and we save in another array the information regarding the position of the bin frequency interval in its original array.


Plugin Parameters

Program	Polyphonic
Min Frequency	55,0000 Hz
Max Frequency	1760,0000 Hz
Voicing Tolerance	0,2000
Monophonic Noise Filter	0,0000

Figure 3.2: MELODIA parameter window in Sonic Visualiser

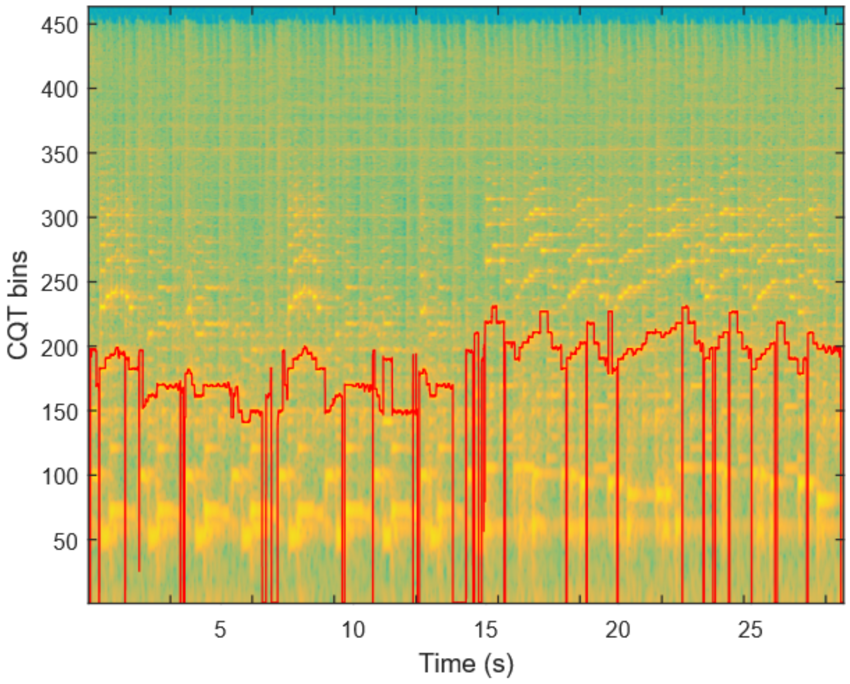


Figure 3.3: Constant-Q Transform of a musical excerpt with the extracted melody overlayed in red

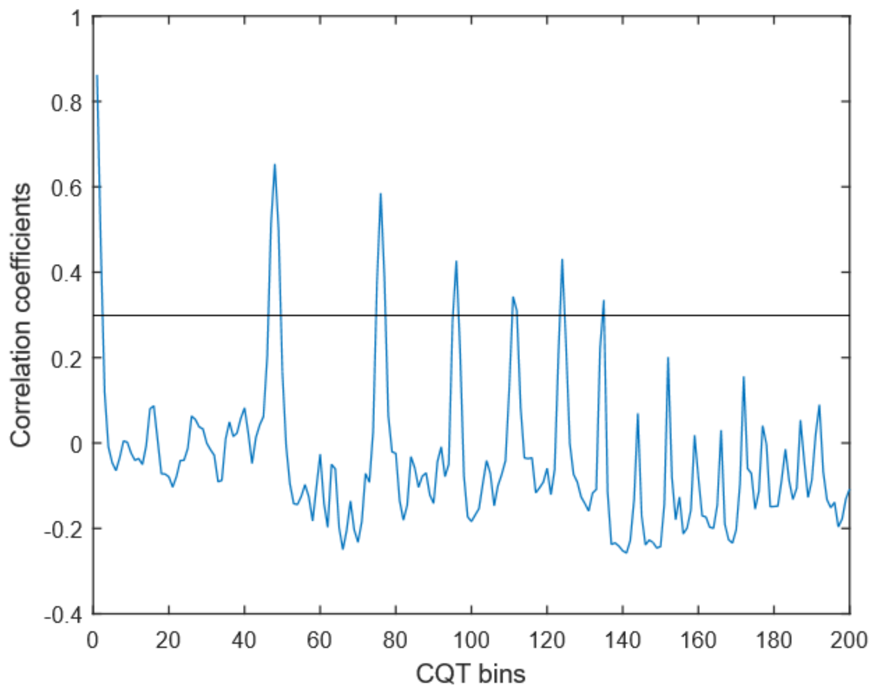


Figure 3.4: Correlation of the main melody with every subsequent bin with the threshold represented as a horizontal line

3.1.4 Identifying the harmonics

When extracting the melody, we had to take into account the existence of harmonics. To include the harmonics as melodic content, we compute a correlation between the melody bins and shifted versions of themselves. The highest correlation along the bins indicates the presence of harmonics. By inspection of figure 3.4, we considered a correlation to correspond to a valid harmonic if its value is equal or above a threshold of 0.3.

3.1.5 Separation

This stage entails creating a mask that keeps the melodic information but erases everything else. To make sure the melody/accompaniment separation is the most accurate, we had to decide how wide the mask contours would be. Either we left the accompaniment virtually untouched at the risk of not completely separating all the melody or we did the opposite and left small vestiges of the accompaniment with the main melody. We opted with the latter approach and by zooming in the CQT matrix, as show in figure 3.5, we concluded that a width of 4 bins up and 4 down would give us a well-balanced result.

The resulting mask is processed so as to narrow the number of bins necessary to cover each subsequent harmonic. For this we chose to neglect any value of the mask that fell under the -120dB threshold, as these would be too low volume to be deemed significant. After both masks

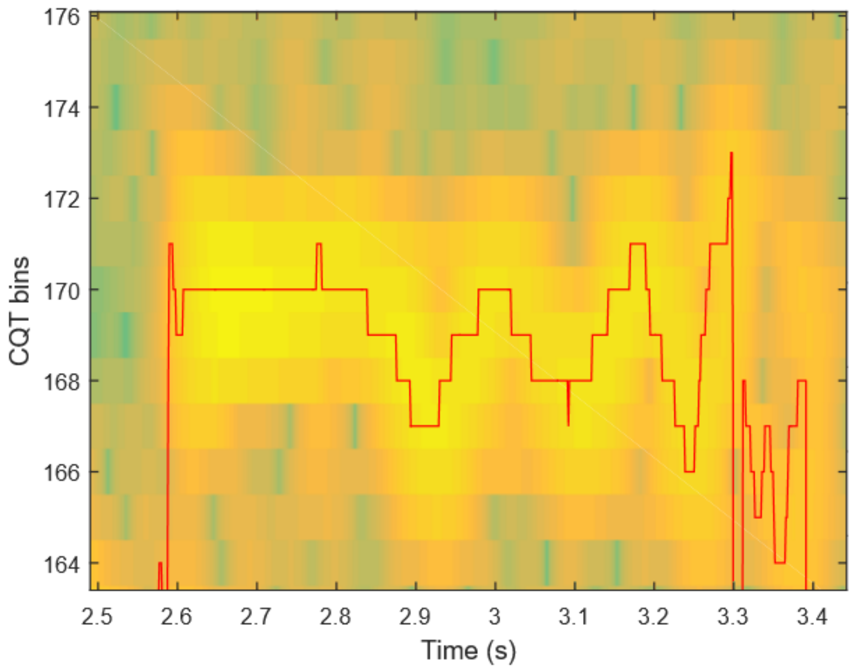


Figure 3.5: Zoomed in veiw of the CQT matrix with the extracted melody overlayed in red

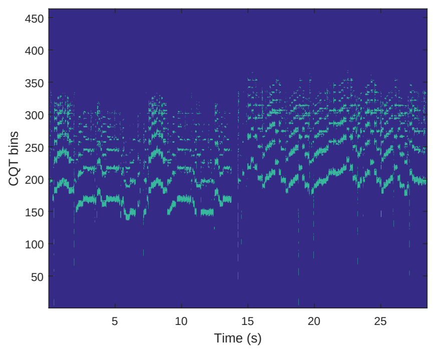
were created, they were simply applied to the CQT, resulting in two matrices: one containing just the melodic content extracted as described above (figure 3.6a), and another containing the remainder of the musical content of the audio file (figure 3.6b).

3.1.6 Pitch Shifting

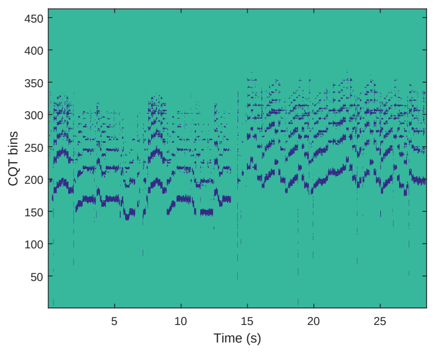
The next stage consists of using the pitch shifting tool [4] to manipulate the specific note selected. The number of semitones chosen for the shift is multiplied by 4 to obtain the correspondent number of bins. The whole file is shifted by that number of bins and we then take the section of the resulting matrix which corresponds to the selected note and replace it in the original matrix to make it so that only the note in question is the one that changed. The reason we do this is because one of the parameters the pitch shifting function requires is the length of the array, which is obtained automatically when calculating the CQT and is always constant for a given file, as it corresponds to the entire length of the matrix. Thus, when using only the selected note as the input of the pitch shifting function, the results would not be the expected given that the parameters would be incoherent.

3.1.7 Reconstruction and Inverse Constant-Q Transform

The final stage is the reconstruction of the signal. In this stage, we put the shifted melody and the original accompaniment back together. In order to do this, we must first delete the part of the



(a) Untreated melody mask



(b) Accompaniment mask

Figure 3.6: Masks used to separated the melody from the accompaniment

accompaniment where the shifted notes will be placed, as seen in figure 3.7, to avoid overlapping the two signals. Next, we add the two matrices together and apply the inverse CQT.

3.2 Interface

Interactivity is a big part of this project and as such a user interface was developed. The aim of this interface is to allow the user to preform various actions of melodic manipulation through a simple and straightforward design. We kept the number of buttons and other elements to a bare minimum not to clutter the layout and keep a focused user experience. The interface is presented in figure 3.8.

3.2.1 Overview

The audio files used in the system are run through MELODIA beforehand to have their melody contour computed.

The 'Open file...' button is used to load one of the prepared files to the system. This action will automatically trigger the calculation of the CQT of the selected file, as well as the quantization of the melody and its separation from the accompaniment, including the harmonics. This results in two matrices: The Melody Matrix and the Accompaniment Matrix. The Melody Matrix is displayed on the screen then the user can use the 'Select Note' button to choose which note is going to be shifted. This button requests two values to indicate the beginning and end of the note, in other words, it asks the user to select a range of columns from the Melody Matrix containing just the desired note.

From this moment, the user can choose what shift to apply. The most straight forward way is to select the shift using the vertical slider. This slider allows for integer values between -11 and +11, corresponding to the number of semitones to shift. The 'all same notes' checkbox allows the user to apply the same shift to all instances of the selected note, as detailed in section 3.2.2. By checking the 'shift everything' box before selecting any note and using the slider, the user is

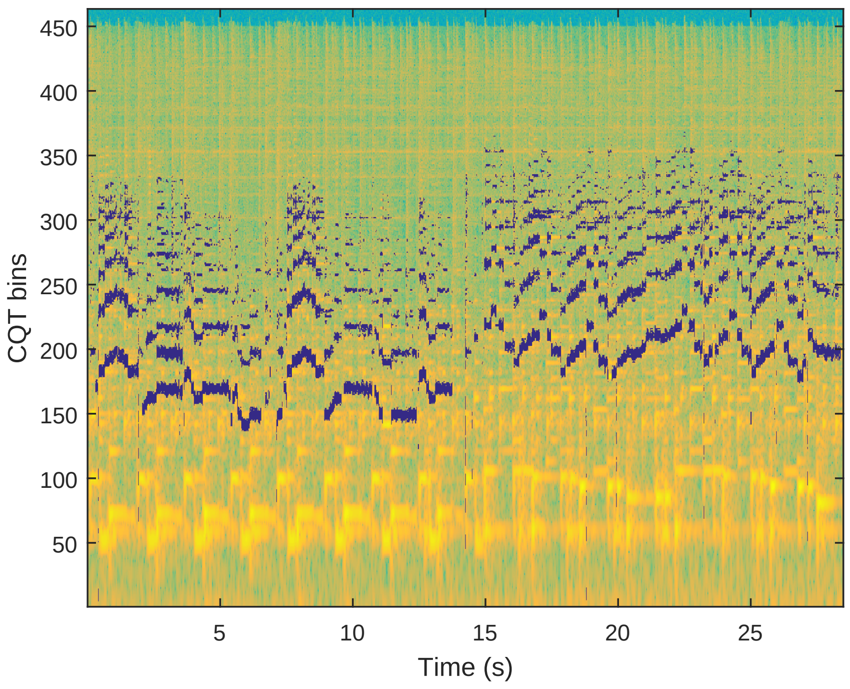


Figure 3.7: Accompaniment with a gap where the shifted melody will be

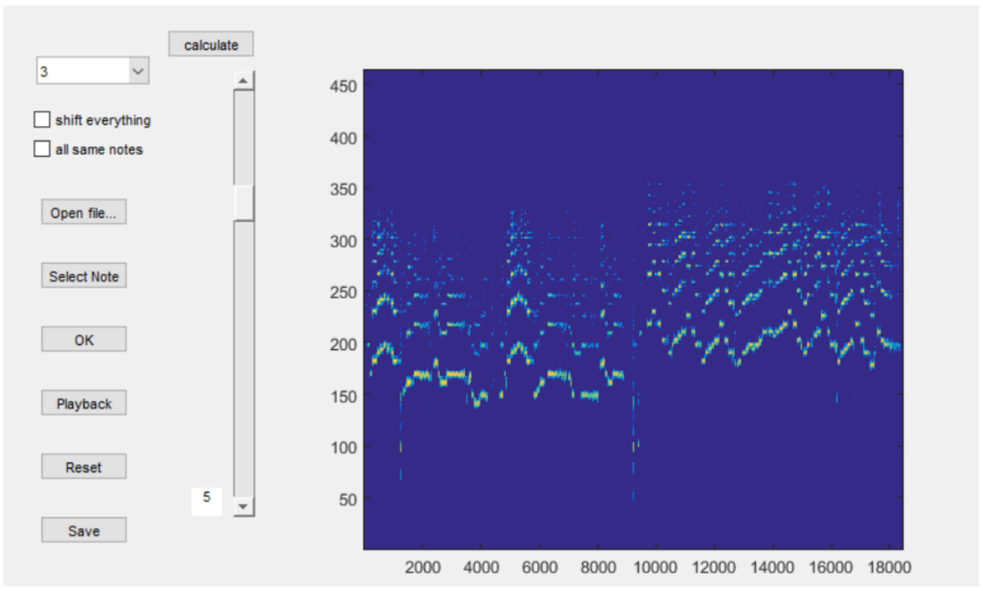


Figure 3.8: User Interface

able to shift the entire melody. Another option is to take a recommended shift from the drop-down menu, which will be covered in section 3.2.3.

The 'OK' button triggers the pitch shift function on the selected note as described in section 3.1.6 using the selected number of semitones. Several pitch shifts can be applied in succession to a single note.

The 'Reset' button reverses every change made since the file was loaded. When either the 'Save' or 'Playback' button is pressed, the reconstruction of the signal is executed and the inverse CQT is applied. The 'Save' button allows the user to save the current changes as a .wav file. Using the 'Playback' button, the user can listen to changes made thus far.

3.2.2 Shifting all instances of the same note

To apply the same shift to every occurrence of a given note, we calculate the mean CQT value and compute its correlation with all the remainder of the Melody Matrix. Its mean value is calculated so that any inaccuracies in the manual selection of the note can be discarded. The correlation is done along the line of the Melody Matrix and allows us to know if there are more of same notes and where they are located. The peaks of figure 3.9 indicate the existence of notes similar to the selection and we used a threshold of 0.6 to delineate which notes are the same as the user selected note. For every note within this threshold, we apply the pitch shift function using the same pitch shift.

3.2.3 Recommending Shifts

To calculate the recommended shifts, the user presses the 'calculate' button. We limit our recommendations to notes that are already present in the main melody, as it assures the shifted notes remain in key, also favouring smaller shifts over larger ones. To do this, we start by calculating the mean value of the selection, similar to what was done to apply the same shift to all instances of the same note, and we also calculate the mean value for the entire melody to know which notes appear and how often they occur (figure 3.10). We used the threshold of 0.006 to define which of the notes present in the melody are also possible shift recommendations. The lower this threshold, the more options will be suggested. However, if we have many options, some of them are bound to not be good shift candidates. For each of these notes the pitch shift S_i is calculated according to

$$S_i = \frac{(n - p_i)}{4}, \tag{3.3}$$

where n is the selected note and p_i, with $i = 1, 2, \ldots$, are the shift candidates. Then, these values are sorted according to how frequent the resulting notes are and the top five values are the ones displayed for the user to select, thus the user is given a ranked set of possible pitch shifts to choose from.

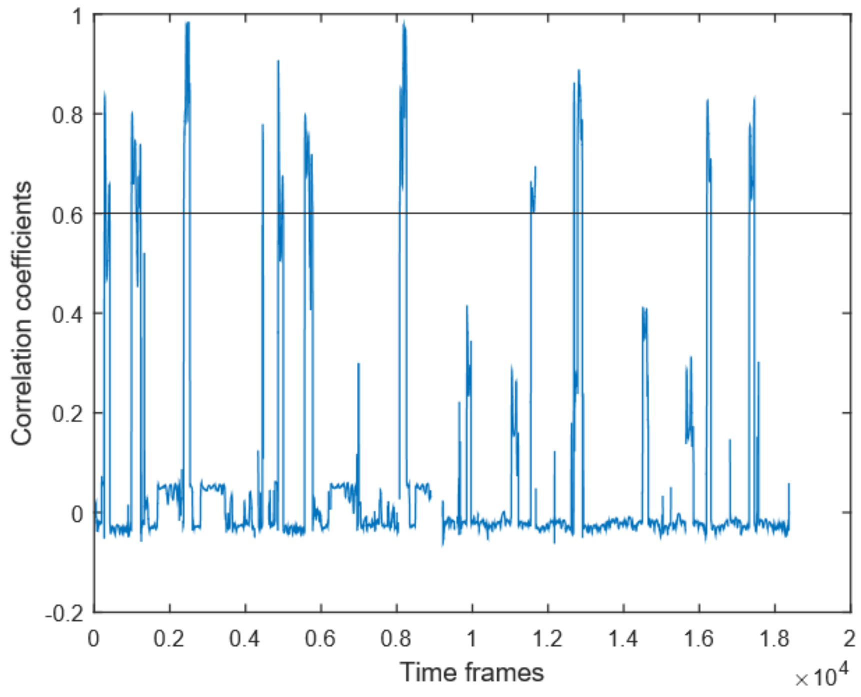


Figure 3.9: Correlation of the selected note with the rest of matrix line. The threshold of 0.6 is indicated by the black horizontal line

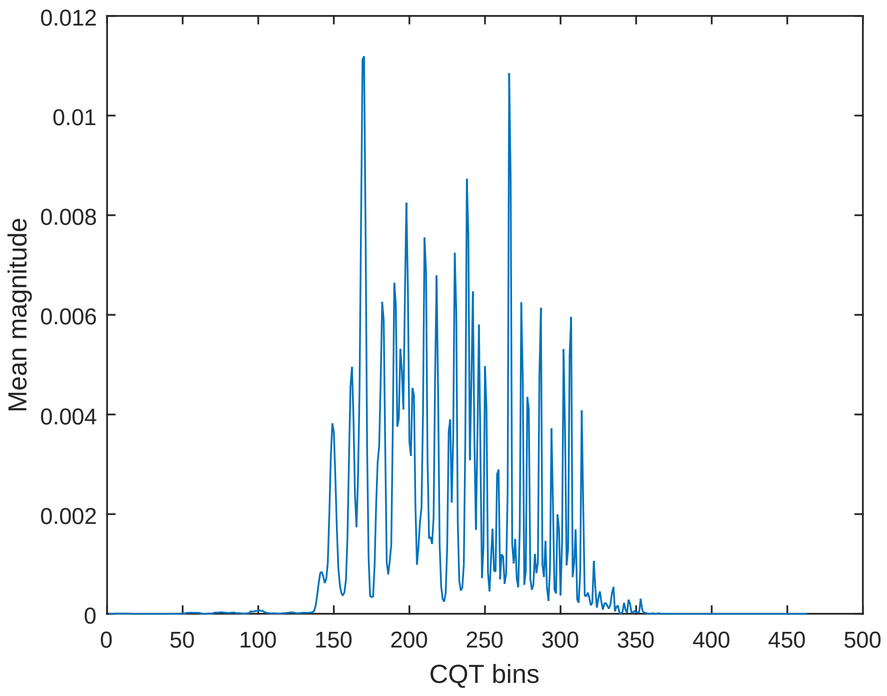


Figure 3.10: Mean melody plot

3.3 Summary

In this chapter we presented the different stages that constituted the developmental process of this project. First the back-end portion of the project, from melody extraction with MELODIA to the application of the CQT and the pitch shift in MATLAB. Then we described the graphic user interface and all its different functions, including changing all instances of a selected notes and recommending pitch shifts.

Chapter 4

Evaluation

In this chapter we discuss the approach taken to evaluate the system and its results. We include a description of how the experiment was designed and conducted and detail what tasks the users were asked to perform with the interface and what questions were asked afterwards regarding the experiment.

4.1 Approach

Given that this project aims to develop a tool to perform creative tasks, we faced a difficulty when evaluating the system. The subjective nature of the tasks at hand mean there is no obvious ground truth to which compare the result obtain through the use of our system. So, our approach to evaluate the system was a group of simple tasks for the user to perform using the interface. The user was asked to open a file and manipulate some of its notes, first using a recommended shift and applying it to all instances of a given note and then apply a simple shift on a different note using the slider and finally listen to the result. We included the document with these tasks in the appendix A.

After the experiment was concluded, users were asked to answer a short survey about the quality of the result and their user experience. The first three questions were regarding background information about the users. We asked if they had musical training, if they have experience in music processing and if they have experience when dealing with music editing software. In the following five questions, users were asked to rank on a scale from 1 to 6 how difficult they found the instructions to be followed, how good was the sound quality of the result, how responsive was the system, how much they enjoyed using the system and how interested they were in using the system again. The final question was a comment box for users to leave suggestions for improvements of the system if they chose to.

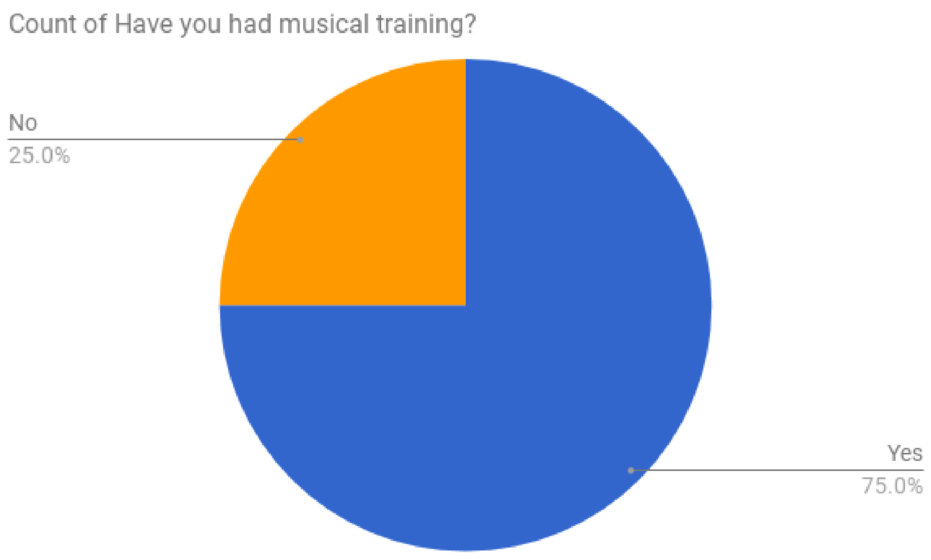


Figure 4.1: Question 1: Have you had musical training?

4.2 Results

The test was conducted under our supervision with twelve volunteer test subjects, who took the test in a quiet environment with good quality headphones. The tasks were the same for every person, using the same audio file and making the same pitch shifts and the full test ran at about five to ten minutes each.

Figures 4.1, 4.2, 4.3 show the distribution of the answers for the first three questions. Of those surveyed, 75% had musical training, 58.3% have experience in music processing and 75% have experience with some sort of music editing software. The results obtained from the survey show a generally positive assessment of the system. Most of the surveyed did not find it too difficult to navigate through the tasks required, with only two answers ranging from moderately difficult to difficult (figure 4.4). The sound quality of the end result was met with a very positive response, with half the users classifying the quality as good and the other half classifying it as very good, as shown in figure 4.5. Regarding the system's responsiveness, as seen in figure 4.6, a third of the users gave it the maximum ranking of six, while another third gave it a 4, indicating moderately responsive. Figure 4.7 shows that more than half the users (58.3%) considered their experience enjoyable, 25% also enjoyed using the system but only moderately so. Most users stated they are interested in using the system again, 25% being very interested, as can be seen in figure 4.8. The most common suggestion we received was the addition of some sort of way to provide the user with feedback regard their and the algorithm's actions, letting them know if a task was completed or if it is being computed. Another common suggestion was the improvement of the graphical user interface and make it more appealing from an aesthetic point of view.

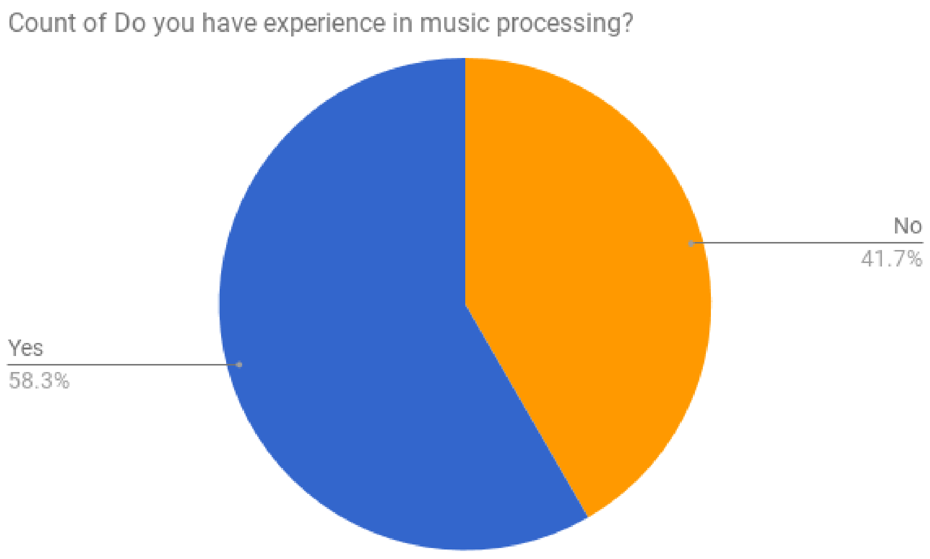


Figure 4.2: Question 2: Do you have experience in music processing?

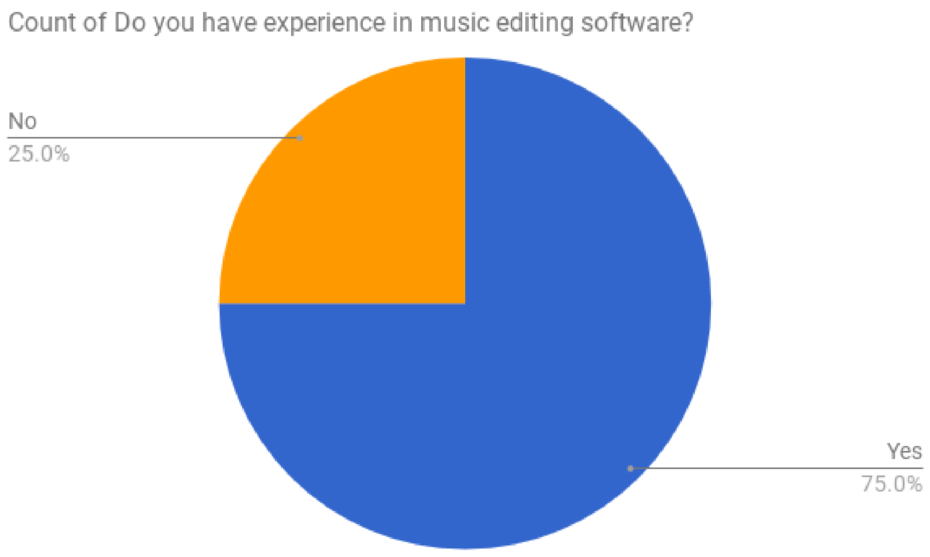


Figure 4.3: Question 3: Do you have experience in music editing software?

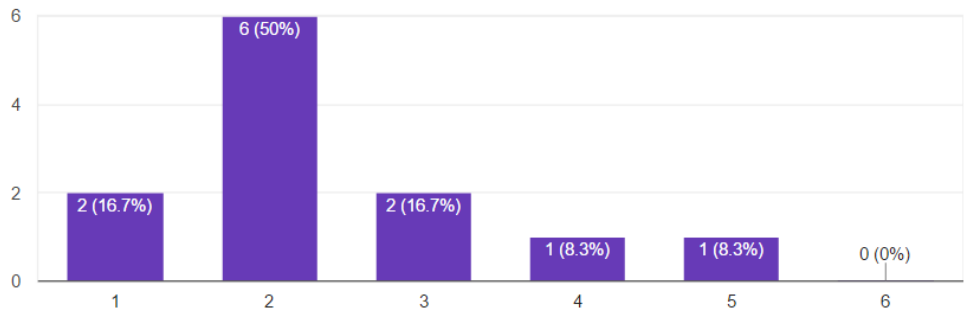


Figure 4.4: Question 4: How difficult was it to follow the instructions? 1 - very easy; 2 - easy; 3 - moderately easy; 4 - moderately difficult; 5 - difficult; 6 - very difficult

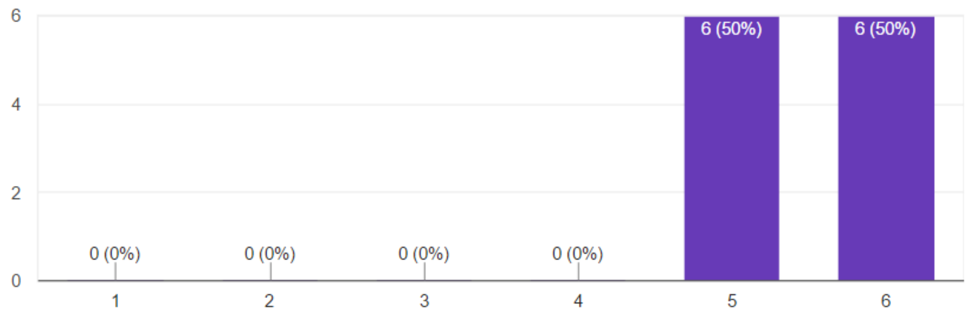


Figure 4.5: Question 5: How good was the sound quality of the result? 1 - very bad; 2 - bad; 3 - average; 4 - moderately good; 5 - good; 6 - very good

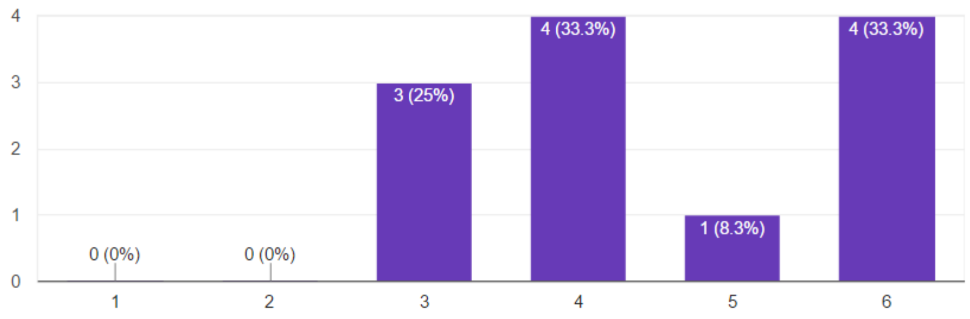


Figure 4.6: Question 6: How responsive was the system? 1 - unresponsive; 2 - fairly unresponsive; 3 - not very responsive ; 4 - moderately responsive; 5 - responsive; 6 - very responsive

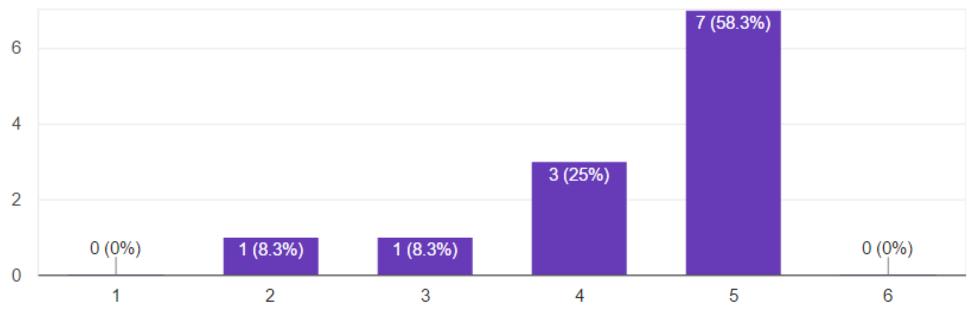


Figure 4.7: Question 7: How enjoyable using the system is? 1 - not enjoyable; 2 - not very enjoyable; 3 - somewhat enjoyable; 4 - moderately enjoyable; 5 - enjoyable; 6 - very enjoyable

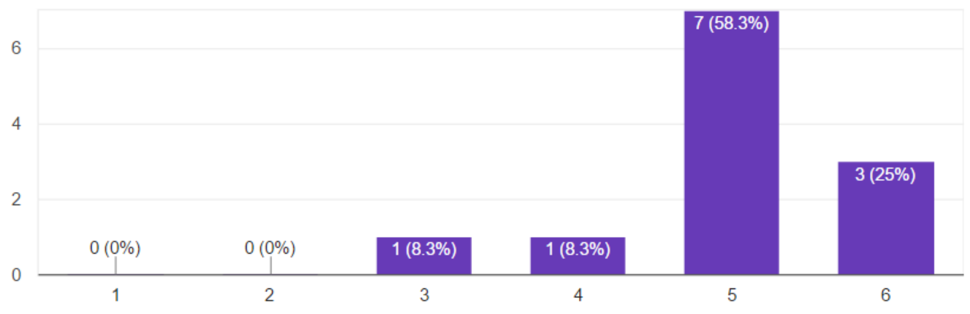


Figure 4.8: Question 8: How interested you would be to use the system again? 1 - not interested; 2 - not very interested; 3 - somewhat interested; 4 - moderately interested; 5 - interested; 6 - very interested

4.3 Summary

This chapter covered the process of evaluating the system. When faced with the difficulty of evaluating creative tasks, we devised a way for users to test the system and assess the quality of the results. The results obtained from the survey indicated that the most crucial objective of the project (sound quality of the result) was achieved. We were also provided with feedback, which will serve as guidelines for future improvements.

Chapter 5

Conclusion and Future Work

This dissertation proposed a system with which users can manipulate the melody of polyphonic audio recordings. The approach taken was to combine existing melody extraction software with a CQT and pitch shifting toolbox for MATLAB and create an environment to apply pitch shifts only to the extracted melody, as opposed to the entire audio file. Via a user-based evaluation of the system, we received largely positive feedback in terms of the resulting audio quality and enjoyment of experimenting with the interface. In the following sections, we reflect on the properties of the melody manipulation system in light of the results obtained, and identify several promising areas for future work.

5.1 Discussion of the Results

While positive feedback was provided by the users who participated in the evaluation, indicating that high quality audio results are possible, we identified a critical dependency in the processing pipeline. At a fundamental level, the performance of this melody manipulation system is only as good as the melody estimation itself. Therefore if there are errors in the initial extraction of melody, e.g. missed notes, or pitch errors, then these will be naturally carried into the melody manipulation stage, thus adversely affecting performance. To circumvent this issue required extensive parameterisation and testing of the MELODIA system in order to interactively discover per-excerpt settings which gave the best subjective results - which in many cases were not the default MELODIA parameters. In the current implementation this melody extraction pre-processing was not exposed to the user, and could be considered too complex for an end-user without technical expertise in audio processing. A second important issue related to the melody extraction concerns the choice of content itself. Which is to say, even after an exhaustive search of parameters, some parts of the melody may still not be identifiable in a fully automatic fashion. To this end we have verified that MELODIA works better on music that has a highly prominent main melody within the mix. Therefore, within a production context, a more interactive approach involving human annotation of the melody may be required – however this is beyond the scope of the current project, and target end-user. The pitch shifting tool used does create some sound quality imperfections upon reconstruction, due to the fact that when we reconstruct the signal we do not replace the gap the note leaves in its original position when shifted. While this is not noticeable when pitch shifting a few isolated notes, the adverse effect this causes can be heard when many notes, or the entire file, are shifted a great amount.

The feedback to the system, as provided by the conducted survey, indicates a success when it comes to the quality of the transformation and how it does not disrupt the original sound quality of the file. This means we were able to develop a functional melody manipulation tool with the proposed method, however further experimentation is required on a wider set of musical material to more deeply understand the types of artifacts that can occur, and hence how to compensate for them. From the answers provided to the first three questions of the survey, regarding the background of the test subjects – which indicated generally high familiarity with music processing tasks, we may speculate that our user group is somewhat skewed and not fully representative of general users. The system was also able to generate interest in the experiment participants to go back and utilize it again in the future. Possible improvements to the interface will be made in a design environment other than MATLAB, to allow for a wider range of design options and capabilities as well as being more responsive.

5.2 Future Work

As future work, we expect to improve the model used to recommend shifts so that it makes more musically informed suggestions rather than relying on a histogram of the most prominent pitches in the recording. When analysing entire pieces of music, it would be important to allow for the fact that the harmony can change between different sections, and therefore to make a local assessment of the most promising pitch shifts to suggest. In addition we also intend to optimize the code so that it runs more efficiently, e.g. by re-implementing the system in a compiled language such as C++. Another improvement we propose is the addition of system functionality to allow different types of musical manipulations, such as temporal transformations and dynamic transformations. The former refers to changing note durations, e.g., making notes last more or less than their original length. The latter is the alteration of intensity, making notes sound louder or quieter or, in musical technical terms, more *forte* or *piano*, respectively.

To make sure the quality of the system is up to par with the industry standards, we recognize the importance of performing a comparative evaluation with existing systems, such a SPEAR and Melodyne. This way we can further validate the method developed for transforming the melody of audio recordings. We can also test the quality of the melody separation via the of multi-track data, (where an isolated stem of the melody can be obtained) and then use standard audio separation metrics such as signal to noise, distortion and interference ratios.

Being able to integrate the melody estimation as part of our system and providing users an intuitive way to set the parameters to control the melody extraction is another of the ways in which we can improve this system.

Finally, an interesting creative application for this project would be to provide the ability to not only manipulate an individual song's main melody, but also to take part of one song and insert it in another song, i.e. take the main melody of one song and place it over the accompaniment track of another song.

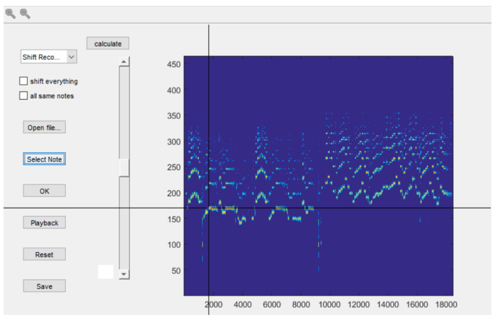
Appendix A

Experiment

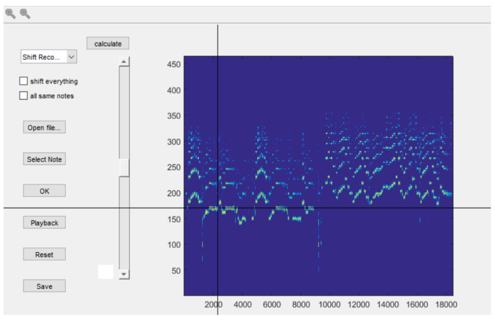
Below we present the tasks user were asked to perform on our interface as the experiment to evaluate the system.

A.1 Tasks

1. Load the file 'take-five1.wav' using the 'Open file. . . ' button.
2. Press the 'Select note' button and select a note by first pressing at the beginning of the note, as indicated in figure A.1a, and then at the end of the same note as seen in figure A.1b.
3. Press the 'calculate' button on the top to compute the recommended shifts.
4. Use the drop-down menu to select the number 3 as seen in figure A.2.
5. Check the 'all same notes' box and press OK (this may take a while. . .).
6. Use 'Select note' like in step 2 and select a different note as seen in figure A.3.
7. Uncheck the 'all same notes' box.
8. Use the slider to select the shift '5'. Do this by sliding the cursor up and seeing the value in the text box at the bottom as seen in figure A.4.
9. Press OK.
10. Press the 'Playback' button to listen to the result.



(a) Beginning of a note



(b) Ending of a note

Figure A.1: Selecting a note

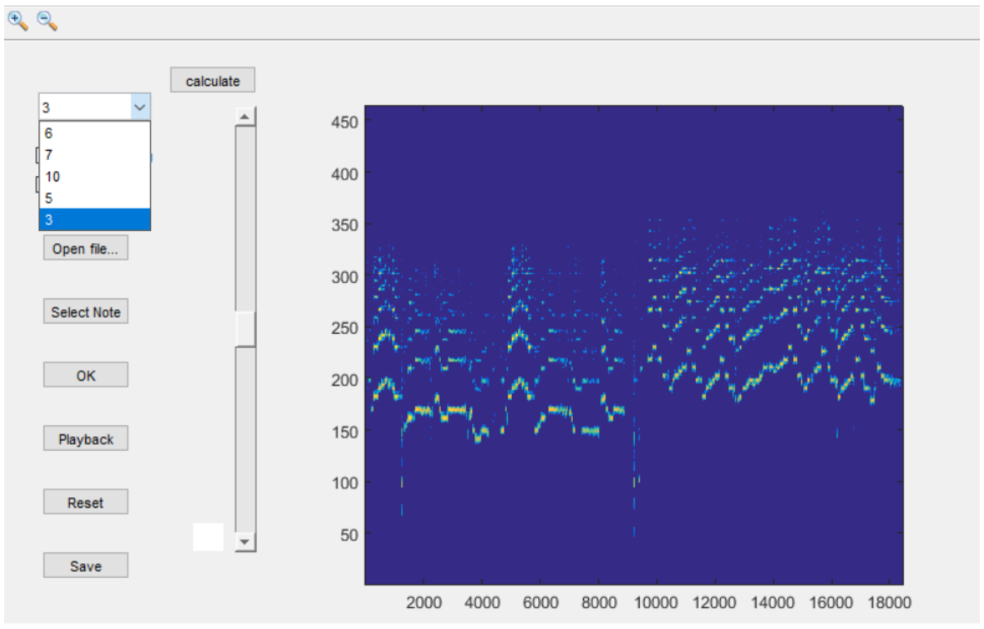


Figure A.2: Using the drop down menu

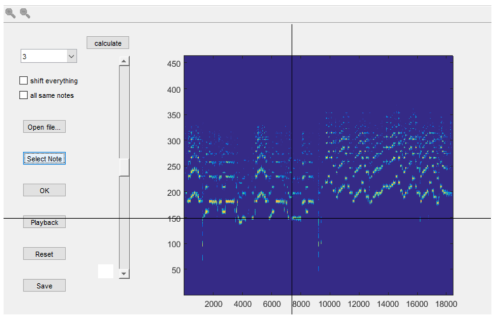
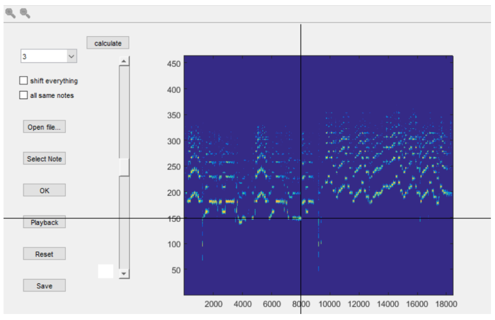



Figure A.3: Selecting another note

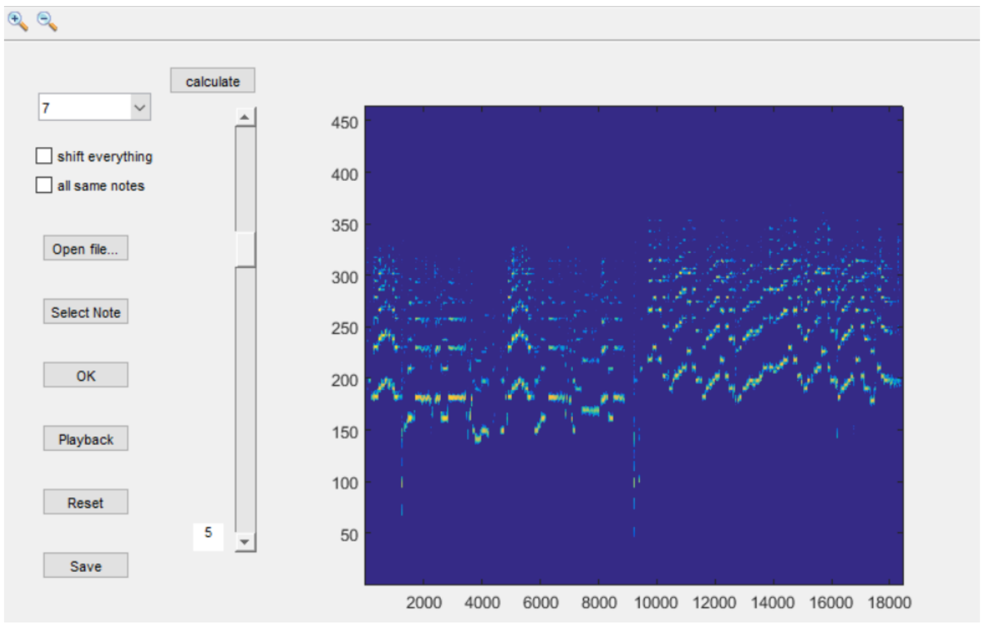


Figure A.4: Slider moved up to the "5" position

References

[1] Christian Schörkhuber, Anssi Klapuri, and Alois Sontacchi. Audio pitch shifting using the constant-q transform. *Journal of the Audio Engineering Society*, 61(7/8):562–572, 2013.

[2] Judith C Brown. Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

[3] Justin Salamon and Emilia Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1759–1770, 2012.

[4] Christian Schörkhuber and Anssi Klapuri. Constant-q transform toolbox for music processing. In *7th Sound and Music Computing Conference, Barcelona, Spain*, pages 3–64, 2010.

[5] Joe Futrelle and J Stephen Downie. Interdisciplinary research issues in music information retrieval: Ismir 2000–2002. *Journal of New Music Research*, 32(2):121–131, 2003.

[6] J Stephen Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[7] Daniel PW Ellis and Graham E Poliner. Identifying cover songs' with chroma features and dynamic programming beat tracking. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–1429–IV–1432. IEEE, 2007.

[8] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[9] Daniel PW Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[10] Yoav Medan, Eyal Yair, and Dan Chazan. Super resolution pitch determination of speech signals. *IEEE transactions on signal processing*, 39(1):40–48, 1991.

[11] David Talkin. A robust algorithm for pitch tracking (RAPT). *Speech coding and synthesis*, 495:518, 1995.

[12] Ernst Terhardt. Calculating virtual pitch. *Hearing research*, 1(2):155–182, 1979.

[13] Ernst Terhardt, Gerhard Stoll, and Manfred Seewann. Algorithm for extraction of pitch and pitch salience from complex tonal signals. *The Journal of the Acoustical Society of America*, 71(3):679–688, 1982.

[14] A Michael Noll. Cepstrum pitch determination. *The Journal of the Acoustical Society of America*, 41(2):293–309, 1967.

[15] Robert C Maher and James W Beauchamp. Fundamental frequency estimation of musical signals using a two-way mismatch procedure. *The Journal of the Acoustical Society of America*, 95(4):2254–2263, 1994.

[16] Martin Piszczalski and Bernard A Galler. Predicting musical pitch from component frequency ratios. *The Journal of the Acoustical Society of America*, 66(3):710–720, 1979.

[17] Pablo Cancela. Tracking melody in polyphonic audio. mirex 2008. *Proc. of Music Information Retrieval Evaluation eXchange*, 2008.

[18] Masataka Goto. A real-time music-scene-description system: Predominant-f0 estimation for detecting melody and bass lines in real-world audio signals. *Speech Communication*, 43(4):311–329, 2004.

[19] Karin Dressler. Sinusoidal extraction using an efficient implementation of a multi-resolution fft. In *Proc. of 9th Int. Conf. on Digital Audio Effects (DAFx-06)*, pages 247–252, 2006.

[20] Chao-Ling Hsu and Jyh-Shing Roger Jang. Singing pitch extraction by voice vibrato/tremolo estimation and instrument partial deletion. In *ISMIR*, pages 525–530, 2010.

[21] Tzu-Chun Yeh, Ming-Ju Wu, Jyh-Shing Roger Jang, Wei-Lun Chang, and I-Bin Liao. A hybrid approach to singing pitch extraction based on trend estimation and hidden markov models. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 457–460. IEEE, 2012.

[22] Matija Marolt. On finding melodic lines in audio recordings. In *Proceedings of the International Conference on Digital Audio Effects*, pages 217–221, 2004.

[23] Matti P Ryynänen and Anssi P Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32(3):72–86, 2008.

[24] Vishweshwara Rao and Preeti Rao. Vocal melody extraction in the presence of pitched accompaniment in polyphonic music. *IEEE transactions on audio, speech, and language processing*, 18(8):2145–2154, 2010.

[25] Karin Dressler. An auditory streaming approach for melody extraction from polyphonic music. In *ISMIR*, pages 19–24, 2011.

[26] Jean-Louis Durrieu. *Automatic transcription and separation of the main melody in polyphonic music signals*. PhD thesis, Ecole nationale supérieure des telecommunications-ENST, 2010. https://tel.archives-ouvertes.fr/tel-00560018.

[27] Laurent Benaroya, Frédéric Bimbot, and Rémi Gribonval. Audio source separation with a single sensor. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):191–199, 2006.

[28] Hideyuki Tachibana, Takuma Ono, Nobutaka Ono, and Shigeki Sagayama. Melody line estimation in homophonic music audio signals based on temporal-variability of melodic source. In *Acoustics speech and signal processing (ICASSP), 2010 IEEE international conference on*, pages 425–428. IEEE, 2010.

[29] Nobutaka Ono, Kenichi Miyamoto, Hirokazu Kameoka, Jonathan Le Roux, Yuuki Uchiyama, Emiru Tsunoo, Takuya Nishimoto, and Shigeki Sagayama. Harmonic and percussive sound separation and its application to mir-related tasks. In *Advances in Music Information Retrieval*, pages 213–236. Springer, 2010.

[30] Justin Salamon, Emilia Gómez, Daniel PW Ellis, and Gaël Richard. Melody extraction from polyphonic music signals: Approaches, applications, and challenges. *IEEE Signal Processing Magazine*, 31(2):118–134, 2014.

[31] Emmanuel Ravelli, Mark Sandler, and Juan P Bello. Fast implementation for non-linear time-scaling of stereo signals. In *Proceedings of the Digital Audio Effects (DAFx) Conference*, pages 182–185, 2005.

[32] Charles Dodge and Thomas A Jerse. *Computer music: synthesis, composition and performance*. Macmillan Library Reference, 1997.

[33] Tristan Jehan. Event-synchronous music analysis/synthesis. In *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFx-04)*, pages 361–366, 2004.

[34] Alan V Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

[35] Judith C Brown and Miller S Puckette. An efficient algorithm for the calculation of a constant q transform. *The Journal of the Acoustical Society of America*, 92(5):2698–2701, 1992.

[36] Gino Angelo Velasco, Nicki Holighaus, Monika Dörfler, and Thomas Grill. Constructing an invertible constant-q transform with non-stationary gabor frames. *Proceedings of DAFX11, Paris*, pages 93–99, 2011.

[37] Michael Klingbeil. Software for spectral analysis, editing, and synthesis. In *ICMC*, pages 107–110, 2005.