UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

QUERY BY HUMMING

A Degree's Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Pau Tur Vallés

In partial fulfilment

of the requirements for the degree in

AUDIOVISUAL SYSTEMS ENGINEERING

Advisor: Antonio Bonafonte

Barcelona, June 2014

Abstract

In this thesis, a Query by Singing/Humming (QbSH) system has been developed. A QbSH system tries to retrieve information of a song given a melody recorded by the user.

The system compares human queries with melodies extracted from audio files. A pitch extraction algorithm has been used to obtain the melodies for both queries and database songs. The preprocessing of the signals turned out to be crucial, and has been deeply studied. The matching step used Dynamic Time Warping, which computes a distance between two signals absorbing tempo variations. Several databases have been built to assess the system. Finally, a complete Graphic User Interface has been programmed to allow the user to analyze the system step by step.

In the end, this thesis contains a thorough experience through the creation of the system which, obtaining competitive results, provides a solid basis for further development.

Resum

En aquesta tesi s'ha desenvolupat un sistema de *Query by Singing/Humming* (QbSH). Aquests sistemes tracten de recuperar informació d'una cançó donada una melodia gravada per l'usuari.

El sistema compara gravacions humanes amb melodies extretes d'arxius d'àudio. S'ha fet servir un algoritme d'extracció del *pitch* per obtindre les melodies de la gravació i de les cançons de la base de dades. El preprocessat dels senyals ha resultat ser crucial, i ha estat estudiat en profunditat. Per la classificació s'ha utilitzat *Dynamic Time Warping*, que calcula la distància entre dos senyals absorbint variacions temporals. Diverses bases de dades s'han construït per avaluar el sistema. Finalment, s'ha programat una completa interfície gràfica per permetre a l'usuari analitzar el sistema pas per pas.

Així, aquesta tesi conté una experiència completa de la creació del sistema que, obtenint resultats competitius, proporciona una base sòlida per futurs desenvolupaments.

Resumen

En esta tesis se ha desarrollado un sistema de *Query by Singing/Humming* (QbSH). Estos sistemas tratan de recuperar información de una canción dada una melodía grabada por el usuario.

El sistema compara grabaciones humanas con melodías extraídas de archivos de audio. Se ha utilizado un algoritmo de extracción del *pitch* para obtener las melodías de la grabación y de las canciones de la base de datos. El preprocesado de las señales ha resultado ser crucial, y ha sido estudiado en profundidad. Para la clasificación se ha utilizado *Dynamic Time Warping*, que calcula la distancia entre dos señales absorbiendo variaciones temporales. Diversas bases de datos se han construido para evaluar el sistema. Finalmente, se ha programado una completa interfaz gráfica para permitir al usuario analizar el sistema paso por paso.

Así, esta tesis contiene una experiencia completa de la creación del sistema que, obteniendo resultados competitivos, proporciona una base sólida para futuros desarrollos.

Agraïments

Als meus pares i al meu germà, perquè m'han espentat quan ha sigut necessari i perquè mai m'ha faltat res.

A la meua parella i als meus amics i família, per fer-ho tot un poc més amè.

I també als meus companys i companyes de la UPC, per compartir rialles i blasfèmies a parts iguals.

Moltes gràcies a tots.

Acknowledgements

I want to thank the supervisor of this thesis, Antonio Bonafonte, for allowing me to do this project and for his help and advice to unblock the development in several occasions.

Also, I want to thank all the professors and the UPC for the knowledge acquired during the last four years.

Revision history and approval record

Revision	Date	Purpose
0	28/06/2014	Document creation
1	07/07/2014	Document revision
2	09/07/2014	Document revision
3	10/07/2014	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Pau Tur	pauturvalles@gmail.com
Antonio Bonafonte	antonio.bonafonte@upc.edu

Written by:		Reviewed and approved by:	
Date	28/06/2014	Date	11/07/2014
Name	Pau Tur	Name	Antonio Bonafonte
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables

1. Introduction

The situation of someone having a melody on his or her mind and not being able to find which song it belongs to is, for sure, familiar for everyone. In this thesis, the wide problem of building a complete Query by Singing/Humming (QbSH) system has been dealt with.

This project has been developed during 5 months. Given the scope of the project, and since it is a final thesis, a complete previous scheduling has been vital.

In following sections, a thorough planning of the whole work is detailed.

1.1. Statement of purpose

The project is carried out at the Departament de Teoria del Senyal i Comunicacions (TSC) at the Universitat Politècnica de Catalunya (UPC) in Barcelona, Spain.

The purpose of this project is to experiment with different techniques in the field of the Music Information Retrieval (MIR) and to build a complete functional prototype of a Query by Singing/Humming system.

The project main goals are:

1. Studying different methods used in a Music Information Retrieval (MIR) application.

2. Learning about the different state-of-the-art techniques for a Query by Singing/Humming system.

3. Building a complete functional Query by Singing/Humming prototype.

4. Experimenting with new options for a Query by Singing/Humming system. Proposing new techniques to improve the performance of the baseline system.

5. Compiling a database and building an experimental framework to assess the different proposals and for further research.

1.2. Requirements and specifications

System requirements:

- The software should retrieve a list with the most similar songs, given a query (in this case, a short segment of a hummed or sung musical piece).
- The performance and behavior of the system should be good, in general terms.
- The software should run normally and smoothly.

- The software should give room for innovation and further improvements.
- The project should provide a developed framework for future research.

System specifications:

- The MRR (weighted recognition rate) should be in the range of the state-of-the-art methods (50-70%).
- The system should be able to retrieve the results list within 15 seconds for a mid-size database.
- The database should contain a large number of songs (over 200).

1.3. Methods and procedures

This thesis develops a project from its beginning, pretending to be usable for future research by providing a baseline system and a Graphic User Interface as a tool to analyze the behaviour of the system.

The main software used for the development is Matlab R2011b. However, an external algorithm (Salamon & Gómez, 2012) has been used in the pitch extraction step. This algorithm is further explained in the third chapter of this thesis. Therefore, and taking into account that the algorithm is available as a VAMP plug-in, an external software, Sonic Annotator (Cannam, O. Jewell, Rhodes, Sandler, & d'Inverno, 2010), has been needed to use the plug-in.

1.4. Workplan and Gantt diagram

Work Packages:

Project: Query by Humming	WP ref: 1
Major constituent: study of the topic	Sheet 1 of 1
Short description: initial study of the main state-of-the-art techniques applied in the Query by Humming tasks.	Planned start date: 01/02/2014 Planned end date:18/02/2014
	Start event: T1 End event: T2

Internal task T1: information gathering. Internal task T2: reading and definition of the project.	Deliverables: Oral report, summary of the main methods.	Dates: Weekly

Project: Query by Humming	WP ref: 2	
Major constituent: baseline system	Sheet 1 of 1	
Short description: creation of a baseline functional system.	Planned start date: 18/02/2014 Planned end date: 21/05/2014	
	Start event: T1 End event: T4	
Internal task T1: feature extraction. Obtainment of the pitch sequences. Internal task T2: classification. Distances among the query and the templates of each song in the database. Internal task T3: integration. Construction of a thorough system in the Matlab environment. Internal task T4: evaluation. Database, metrics and results. Study of the current legal framework for the database creation.	Deliverables: Oral reports, written draft.	Dates: Weekly, May 21st.

<table>
<tr><td>Project: Query by Humming</td><td colspan="2">WP ref: 3</td></tr>
<tr><td>Major constituent: enhancement</td><td colspan="2">Sheet 1 of 1</td></tr>
<tr><td rowspan="2">Short description: improvements of the system and Graphic User Interface.</td><td colspan="2">Planned start date: 22/05/2014
Planned end date: 20/06/2014</td></tr>
<tr><td colspan="2">Start event: T1
End event: T3</td></tr>
<tr><td>Internal task T1: technical improvements.

Internal task T2: evaluation and database improvements. Testing alternative options.

Internal task T3: Graphic User Interface. Creation of a user-friendly environment as a showcase of the developed technology.</td><td>Deliverables:
Oral reports, written draft.</td><td>Dates:
Weekly, June 20th.</td></tr>
</table>

<table>
<tr><td>Project: Query by Humming</td><td colspan="2">WP ref: 4</td></tr>
<tr><td>Major constituent: documentation</td><td colspan="2">Sheet 1 of 1</td></tr>
<tr><td rowspan="2">Short description: writing and correction of every document to be delivered.</td><td colspan="2">Planned start date: 05/03/2014
Planned end date: 11/07/2014</td></tr>
<tr><td colspan="2">Start event: T1
End event: T3</td></tr>
<tr><td>Internal task T1: Project proposal and Work Packages.

Internal task T2: Project Critical Review.

Internal task T3: Final Report.</td><td>Deliverables:
Proposal, PCR, FR</td><td>Dates:
March 14th (Proposal), April 30th (PCR), July 11th (FR).</td></tr>
</table>

Milestones:

WP#	Task#	Short title	Milestone / deliverable	Date (week)
1	T1,T2	Definition of the project	28/02/2014	4
2	T1,T2,T3	Baseline system	15/04/2014	15
2,3	T4(2),T2(3)	Database and evaluation	30/05/2014	17
3	T1,T2	Improvements	20/06/2014	20
3	T3	Graphic User Interface	10/06/2014	19
4	T1	Project Proposal	14/03/2014	6
4	T2	PCR	30/04/2014	13
4	T3	FR	11/07/2014	23

Table 1: Project milestones

Updated time plan (Gantt Diagram):

	Nombre	Duracion	Inicio	Terminado	Predecesores
1	Information gathering	3 days	3/02/14 8:00	5/02/14 17:00	
2	Reading and definition of the project	8 days	6/02/14 8:00	17/02/14 17:00	1
12	Feature extraction	10 days	18/02/14 8:00	3/03/14 17:00	
4	Classification	54 days	1/03/14 8:00	15/05/14 17:00	
5	Integration	7 days	16/05/14 8:00	26/05/14 17:00	4;3
6	Evaluation	6 days	27/05/14 8:00	3/06/14 17:00	5
7	Technical improvements	16 days	15/05/14 8:00	5/06/14 17:00	
8	Evaluation improvements	11 days	6/06/14 8:00	20/06/14 17:00	7
9	Graphic User Interface	12 days	25/05/14 7:00	10/06/14 17:00	
10	Project Proposal	8 days	5/03/14 8:00	14/03/14 17:00	
11	Project Critical Review	9 days	18/04/14 8:00	30/04/14 17:00	10
12	Final Report	16 days	20/06/14 8:00	11/07/14 17:00	11

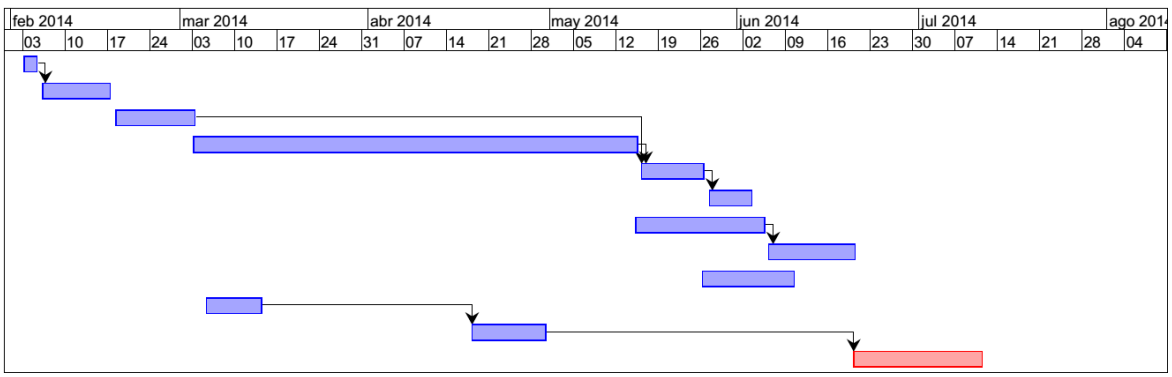

Figure 1: Tasks and Gantt Diagram

1.5. Incidences

Some changes have been made from the initial planning. The improvements have been mainly focused in building a more practical interface instead of testing a higher number of possible preprocessing filters (which is already very high).

Such a change is due to a delay in the preprocessing and matching stages. The output of the pitch extraction algorithm required more processing than expected in order to be suitable for the matching step. This overlapped the baseline system and enhancements work packages and reduced the time for improvements. However, the user interface has been added more functionalities, and alternative tests, such as query vs. query, have been executed as well.

2. State of the art of Query by Singing/Humming systems

As QbSH is a quite recent topic with no universally accepted solution, the proposed solutions vary drastically. The general idea is to *extract features* of the user query, and *match* them with the *reference signals* of the song database. Different approaches are proposed for reference signals, feature extraction, matching, and even assessment of the system.

In this section, most methods applied in these steps are reviewed.

2.1. Reference signals

When developing a QbSH system, the first big decision to be made is the choice of the reference signals, that is to say, the signals from which the database is built. Namely audio files, MIDI files or queries can be used to form the database of the system.

2.1.1. Audio files

Audio files are the typical music files that almost everyone these days has on his or her computer, smartphone, music player or other electronic devices. These files contain audio signals, the real music codified digitally. There are many different audio file extensions depending on the compression, the quality, etc.

When using these files, the melody is extracted from the raw mix of musical instruments that we find in every typical song. Therefore, the melody extraction process comes out as a tough task, since every musical instrument or sound in the song is mixed in the frequency spectrum with the main melody, hindering the identification of the latter.

However, obtaining this kind of files is completely automatic, allowing the database to grow easier and faster, unlike the MIDI files.

Many works with audio files have been done, some of them being (Phiwma & Sanguansat, 2010), (Ito, Kosugi, Makino, & Ito, 2010), (Jain, Jain, Patil, & Basu, 2011), (Song, Park, Yang, Jang, & Lee, 2013), (Jang, Song, Shin, Park, Jang, & Lee, 2011), (Rocamora, Cancela, & Pardo, 2013) and (Salamon, Serrà, & Gómez, 2013).

2.1.2. MIDI

MIDI (Musical Instrument Digital Interface) is a standard protocol of music description and communication which is widely used by most electronic musical instruments and computers.

With MIDI protocol, each musical feature of a note (value, duration, tone, effects...) is represented with numbers. Therefore, MIDI files do not contain an audio signal, but contain digital information to be able to reproduce each note instead.

Inside a MIDI file, information of the different musical instruments is stored in separated tracks. As a consequence, the main melody (generally the vocal part) is contained in just one track. When used in a QbSH system, the goal is to identify the MIDI track containing the main melody, and then extract it.

The predominant representations of songs are studio or live recordings, and MIDI files cannot be automatically derived from audio files. Manual human work is required to create a MIDI file, that is to say, somebody has to transcribe the music into MIDI. Therefore, the biggest limitation for MIDI is that the process is not automatic at all; the first step for the database creation requires a big human effort.

Most research is based on databases built over MIDI files, such as (Antonelli, Rizzi, & del Vescovo, 2010), (Li, Han, Shi, & Li, 2010), (Tsai & Tu, An Efficient Query-by-Singing/Humming System Based on Fast Fourier Transforms of Note Sequences, 2012), (Song, Park, Yang, Jang, & Lee, 2013), (Guo, Wang, Yin, Liu, & Guo, 2012), (Dong & Qi, 2010), (Kotsifakos, Papapetrou, Hollmén, Gunopulos, Athitsos, & Kollios, 2012), (Guo, Wang, Liu, & Guo, 2013) and (Tsai, Tu, & Ma, 2012).

2.1.3. Queries

Finally, short recordings sung by different people can be used to form the database. The variety of singers can strengthen the reliability of the database. Moreover, the melody extraction process is the same both in the database creation step and in the comparison with the query step.

On the other hand, building a good database would require plenty of recordings. As hiring some users to sing every song in the database several times seems unfeasible, the idea with this method would be to train the database with the recordings coming from every user of the system. For instance, the system would implicitly validate whether the results presented to the user are correct or not, and then the query would be used to train the database.

Most research is focused on systems using MIDI or audio files, but some commercial systems using queries as databases are available, as Midomi[1].

2.2. Feature extraction

The feature extraction step is also very important for a QbSH system. Although most systems use the pitch sequence as the main feature, there are several proposals using different features. However, most features are based on the same principle: the evolution of the signal on the frequency domain.

2.2.1. Fundamental frequency and pitch

The fundamental frequency (often noted as f0) is defined as the greatest common divisor of the harmonics in a particular segment of a signal, that is, in its spectrum. Nevertheless, a more perceptual feature is used in the vast majority of QbSH systems: the pitch sequence.

The aim is to find the frequency representation of the main melody of a song. In other words, the main melody is represented as the evolution of a single frequency value, which could be easily identified with the song if heard.

This feature could be said to be the physical representation of the melody of the song. However, the melody extraction process done with typical pitch extraction algorithms is not a simple step. Its reliability depends a lot on the type of signal being used, and there is still a lot of research to be done on this step.

Most pitch extraction algorithms are based on harmonics searching and weighting, and use some complex techniques on the process.

Usually, a logarithmic scale is used to represent the sequence due to the relationship among the musical notes and the physical frequency values. This is the same scale used in the MIDI note sequence (see next point).

2.2.2. MIDI note sequence

A track containing the main melody in a MIDI note sequence is a very similar feature to a pitch sequence. In fact, it can be obtained as a transformation of the pitch sequence. If the pitch sequence could be considered as the physical representation of a melody, the MIDI note sequence could be said to be a musical representation of the melody.

Therefore, its behaviour and usage does not differ much from the pitch sequence case. It is just a different representation.

[1] Midomi – Soundhound (www.midomi.com)

2.2.3. Fast Fourier Transform

Some systems (Tsai, Tu, & Ma, 2012) try to avoid the direct temporal sequence comparison in the matching step. To do so, the note sequences are transformed into Fast Fourier Transform (FFT) vectors. In this way, the problem of matching variable-length sequences is solved, and the matching is faster than with other methods, as simple Euclidean distances can be applied.

However, although results are often good, this method loses the capacity of absorbing tempo variations in the matching step as other methods do. In the end, it is a trade-off situation among better results and faster performance.

2.3. Matching

The matching process finds the likelihood between the query and each song in the database. Therefore, this step aims at computing a numeric value to represent this similarity.

Most QbSH systems these days use Dynamic Time Warping, although some others opt for training-based classifiers or simple distances.

2.3.1. Dynamic Time Warping

In time series analysis, the Dynamic Time Warping algorithm is used to measure the resemblance between two sequences which may vary in time or speed.

The DTW algorithm is widely extended among almost every QbSH system. The main reason for that is the fact that is a very flexible and useful algorithm: it can absorb the most typical imperfections on humans trying to sing a song, being tempo differences the most common error.

2.3.2. Support Vector Machines and Artificial Neural Networks

Support Vector Machines (SVM) is one of the most used and best-performing classifiers. As a result, some researchers have tried to apply it for a QbSH system (Phiwma & Sanguansat, 2010).

However, it has some limitations. The main one is the fact that the features to be classified must have the same dimension. This is a big drawback, since it is difficult to represent the temporal dynamics with a fixed length vector of features.

A system has been developed with SVM. Firstly, it used DTW to compute the distance between the query and each template in the database, and then used this distance vector as the non-variable size vector for the SVM classifier. Consequently, the database cannot

be easily broadened, since the dimension would also change and the SVM classifier would have to be trained again.

In the same paper, Artificial Neural Networks (ANN) are used in the same way as the SVM in a QbSH system, but the results turn out to be worse. Thus, in case of using non-variable size features with this classifiers, SVM is still a better option to use.

2.3.3. Simple distances

Although the general procedure in this step is rather simple, it is only used when the signals to be compared have the same size, that is to say, when the resulting features are the FFT of each signal, as in (Tsai, Tu, & Ma, 2012).

The maths behind this step are quite typical, since it consists in computing the Euclidean distance (or some similar variation) between the values in the FFT vector. Although the combination of FFT features and distance computation on the matching step turns out to be very light computationally, its performance is not the best for a QbSH system.

2.4. Assessment

An assessment step is required to quantify the performance of the QbSH system. Mean Reciprocal Rank is the most extended metric, together with top lists. These two metrics assess almost every QbSH system currently.

2.4.1. Mean Reciprocal Rank

The Mean Reciprocal Rank (MRR) is a great measure of the performance of a QbSH system, since provides a weighted result for the evaluation of a set of queries. The formula to compute the MRR is:

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i}$$

Where N is the number of queries evaluated, and the rank is the position of the correct result in the sorted list. Almost every QbSH system uses this formula to assess the general behaviour of the system. Therefore, a perfect system would provide MRR equal to 1, while a system with MRR equal to 0 would have failed the search for every single query in the test set.

2.4.2. Top-X list

Few QbSH systems use this assessment method. The aim is just to list the first X results, being Top-3, Top-5 and Top-10 the most typical lists. Then, the system counts the number of queries with a correct result returned on the first X positions of the list, so there is no weighting at all, unlike the MRR case.

Being all the different approaches explained, the next section gives a deeper view into the final choices for the QbSH system built in this thesis.

3. Methodology / project development:

The aim of this project is to create a completely functional QbSH system. Through the following points, the creation of the baseline system is reported. First, the reference signals are chosen; then, the features are extracted and preprocessed; finally, the signal is matched with other signals stored in a database. The assessment of the system, however, will be explained in section 4.

3.1. Basic block diagram of the system

The graphic shows the main stages of the QbSH system (Fig. 2). The pitch is extracted from the queries. The preprocessing stage enhances the signal for the matching stage, and then the results are retrieved in the end.

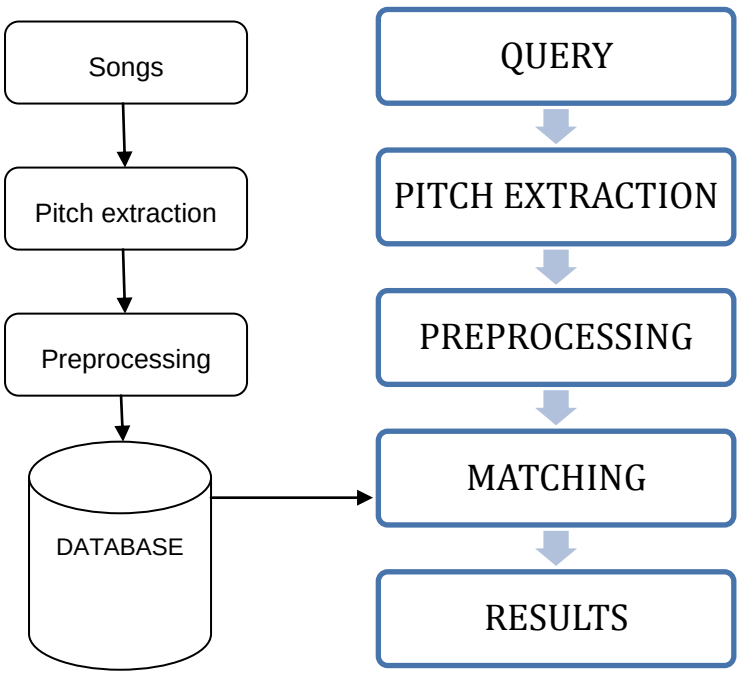

Figure 2: Basic block diagram

3.2. Signal choice and acquisition

As is explained in the second chapter of this thesis, a QbSH system can work with three different file types: MIDI files, audio files, or query recordings. The main advantages and drawbacks are also explained in the second chapter. This is a crucial choice as it influences on the flexibility and performance of the system.

In this system, the database is created from audio files. This decision has been motivated by several factors. First of all, audio files are the easiest file types to obtain among the three possibilities. Therefore, audio files allow a faster and wider update of the database, which seems very important these days, since plenty of songs are daily released. Moreover, systems using audio files are known to obtain the lowest performance, but they also have much research to be done, so it seemed an interesting option.

As a result, creating a database ends being a task of song selection and procurement. The database creation is reviewed in the assessment chapter of this thesis.

Now, the real acquisition process is done for the query, not for the database signals, provided that audio files are being used to build the database. In this system, an audio fragment with a sample frequency of 44100 Hz and 16 bits for sample coding is recorded using a simple laptop microphone.

3.3. Pitch extraction algorithm

Once the query has been recorded, the next step is to extract the pitch, that is, the sequence of frequency values that define the melody of the recording. This step is also applied in the melody extraction step to create the database (Fig. 3).

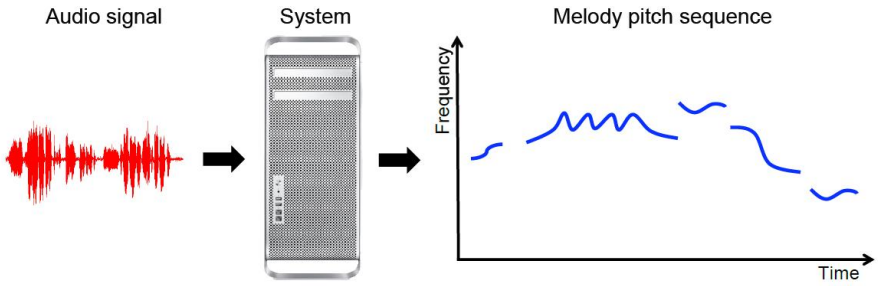

Figure 3: Pitch extraction system (Salamon, Gómez, Ellis, & Richard, 2014)

In this project, a recent pitch extraction algorithm for polyphonic signals has been used. This algorithm, which has one of the best performances for polyphonic signals, was

developed by the Music Technology Group in Universitat Pompeu Fabra (Salamon & Gómez, 2012), and is available as a vamp plugin[2].

3.3.1. Algorithm overview

This pitch extraction algorithm belongs to the group of algorithms known as salience-based methods, which derive an estimation of pitch salience over time and then apply tracking or transition rules to extract the main melody line.

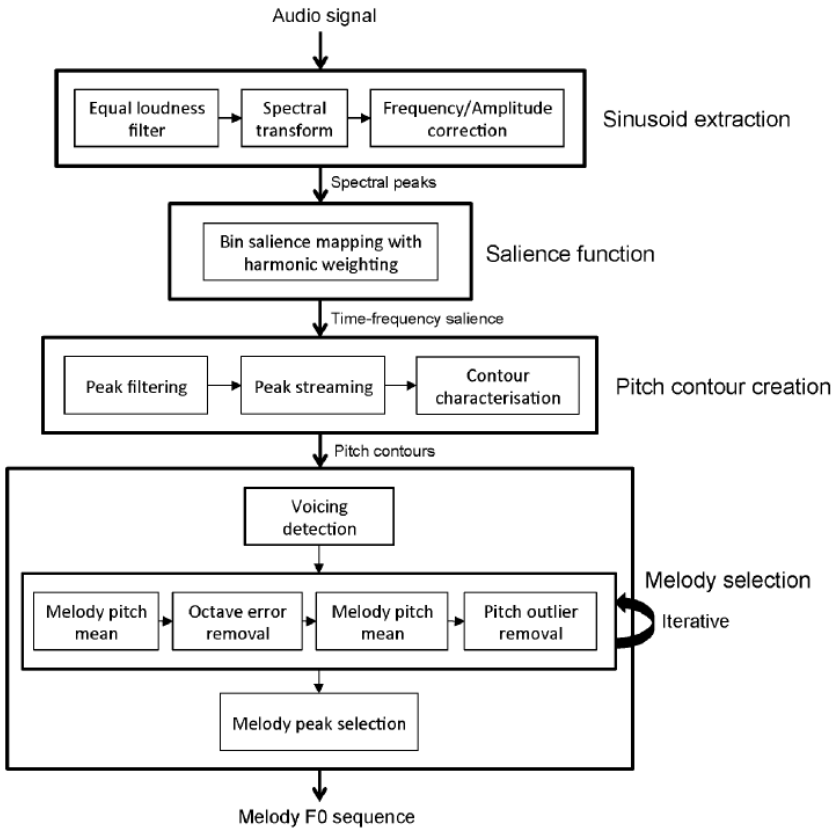

Figure 4: Block diagram of the pitch extraction algorithm (Salamon & Gómez, 2012)

The complete block diagram of the system is shown above (Fig. 4). First, the algorithm applies an equal-loudness filter to the signal in order to emulate the behavior of the human ear, and performs a spectral analysis and correction. Then, the salience function is represented (Fig. 5).

[2] MELODIA vamp plug-in: http://mtg.upf.edu/technologies/melodia

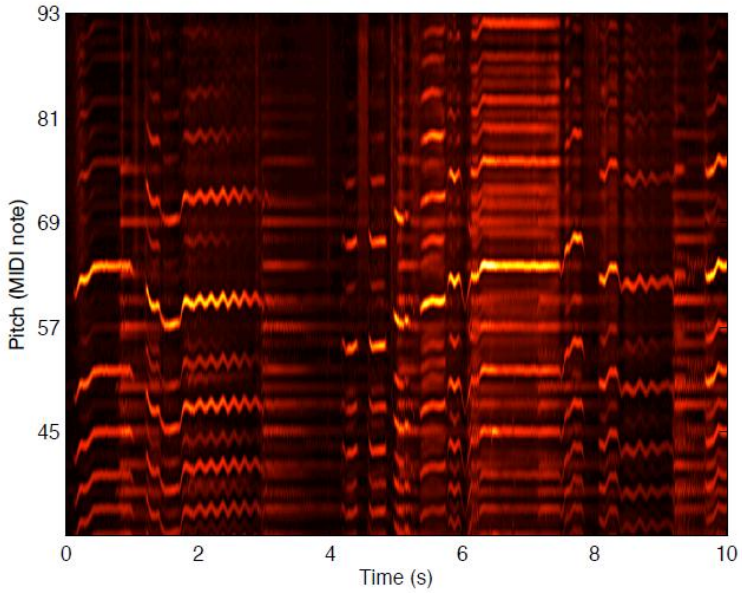

Figure 5: Example of output of the salience function (Salamon, Gómez, Ellis, & Richard, 2014)

The peaks of this function are considered as potential candidates for the melody, so these peaks are properly processed, by means of peak filtering and peak streaming steps, in order to extract the most likely pitch contour. To do so, several features of each potential pitch contour are computed (pitch mean, pitch deviation, contour mean salience, contour salience deviation, length, etc).

Depending on the computed values for these features, the algorithm decides whether the sequence belongs to the main melody to be extracted or belongs to a non-principal melodic sequence (for example, harmonics of the real melodic sequence).

Moreover, an iterative melody selection stage is also applied. In this stage, the melody contour is refined by applying some additional steps, such as voicing detection and removing octave errors and pitch outliers, which are very common. However, not all the errors are automatically removed, so some additional steps are required in the preprocessing stage to improve the signal.

In the end, a sequence of frequencies representing the extracted melody line (the output of the pitch extraction algorithm step) becomes the signal to work with in the following steps.

The following plot shows the output of the pitch extraction algorithm (Fig. 6), which is used within the following points to show the evolution of the signal through the preprocessing step.

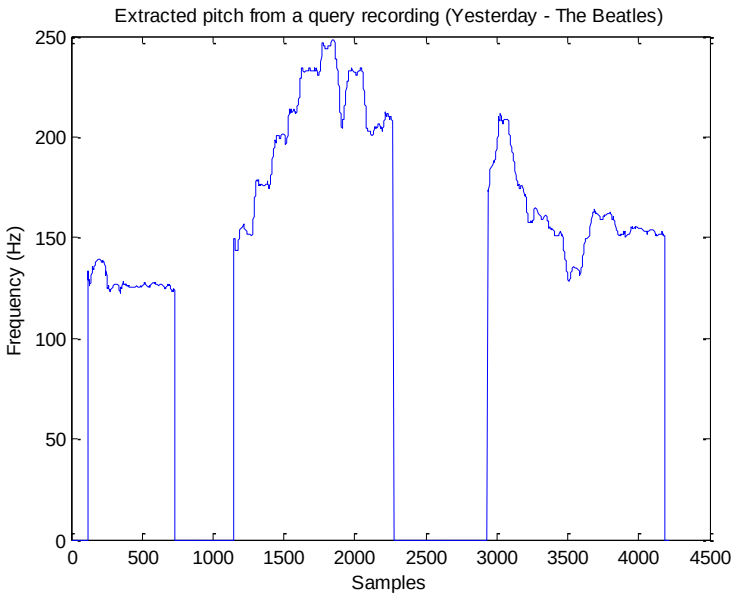

Figure 6: Example of output from the pitch extraction algorithm

The performance of the algorithm depends on the signal on which we are applying the extraction. For music files with a heavy presence of the main melody and soft chords of light instruments, the algorithm behaves quite well. This usually happens with soft jazz and most acoustic songs, for instance. However, in rock songs, where the instrumental load is bigger and more varied, the spectrum becomes very impure, and the algorithm has serious troubles to obtain a good representation of the melodies. The most common errors in this case are skipping notes, selecting wrong notes and selecting wrong octaves. The latter is the only one which can be partially fixed in the preprocessing stage.

In the end, the results obtained by the system depend markedly on the resemblance among the extracted melodies and the real melodies of the songs, provided that the pitch extraction process in the query signal is reasonably accurate. Therefore, the type of music (more precisely, the instrumental load of the song) determines the goodness of the melody extraction process and, consequently, has a great impact on the overall performance of the system.

3.3.2. Alternatives for melody extraction of the query

It is important to remark that this pitch extraction algorithm has been specially designed for polyphonic signals, that is, several sounds being played simultaneously (different notes of the same instrument, different instruments playing their own line, etc.). However, the system uses the same algorithm for the query recordings, which contain a monophonic sequence sung/hummed by a human voice.

The main reason for that is the overall performance of the system. With this algorithm, few notes of the queries can be neglected in the extraction process. As a consequence, an alternative algorithm to extract the query pitch sequence has been tested. In this system, the alternative algorithm is based on the well-known RAPT (Talkin, 1995), which

is a pitch extraction algorithm commonly used in voice processing works. The pitch extraction process with the RAPT algorithm is more complete, detecting every note, than with the MELODIA plug-in. However, the resulting pitch sequence seems to be more inaccurate in general terms, worsening the overall performance of the system.

Therefore, the final implementation of the system uses the MELODIA vamp plug-in for both query pitch extraction and database creation.

3.4. Preprocessing

The preprocessing stage consists in transforming the extracted pitch sequence in order to prepare the signal for the matching step. This block allows many different possibilities and combinations, and is crucial for the final performance of the system. In this system, many different alternatives have been proposed and tested.

One of the most important features in music is the well-timed combination of notes and pauses. In the extracted pitch sequences, the pauses (silence) are represented with a frequency equal to zero. If a user tries to guess a song, the pauses among notes will certainly be as important as the notes themselves. Therefore, it seemed logical to maintain the timing of the pitch sequence by keeping the zero frequencies. However, results turned out to be better when silence has been removed in almost every case.

The plot (Fig. 7) shows the previous signal (Fig. 6) once the silence removal step has been applied.

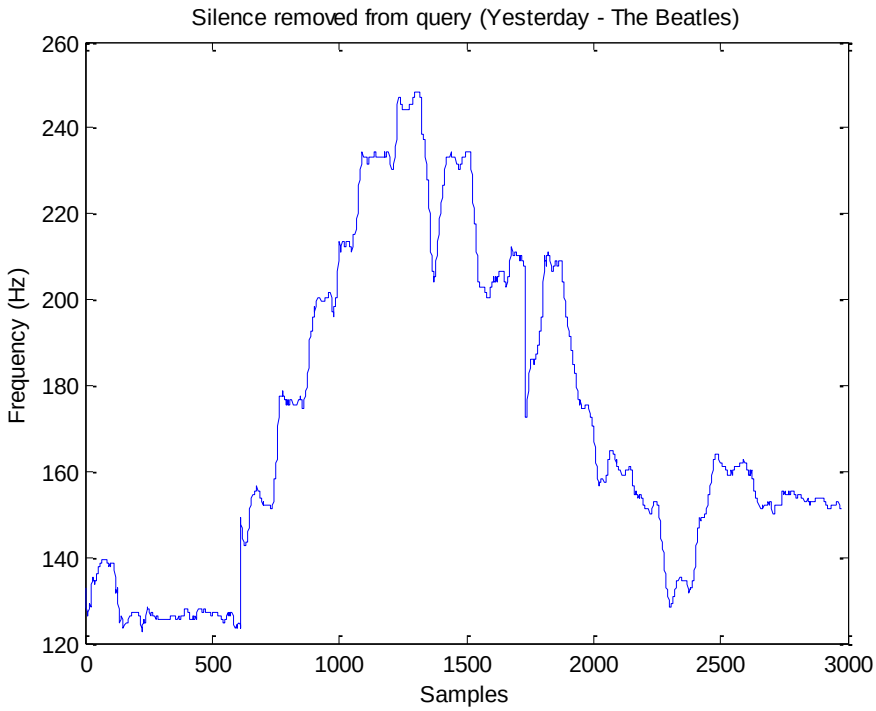

Figure 7: Pitch sequence after applying silence removal

Now, the signal has many little peaks, so the next preprocessing step pretends to smoothen the sequence. To do so, two types of filter have been tested: an average filter, which substitute a sample by the average value of the surrounding samples, and a median filter, which changes the value of a sample by the value in the middle of a sorted list of the surrounding samples (Fig. 8).

When this filters are applied over a signal including silence (which was intended to be removed after this step), some low frequencies appear where silences used to be. Then, the remaining zeros were deleted, but as these low frequencies were still present, they were padded to zero and an interpolation was performed to create a more continuous signal. However, in the final version of the system this step is omitted, as the silence removal is performed firstly obtaining better results.

Visually, the differences between using an average filter and using a median filter are not very noticeable. However, median filters outperformed average filters in most cases. Also, different sizes for the filters have been tested to find the optimum value.

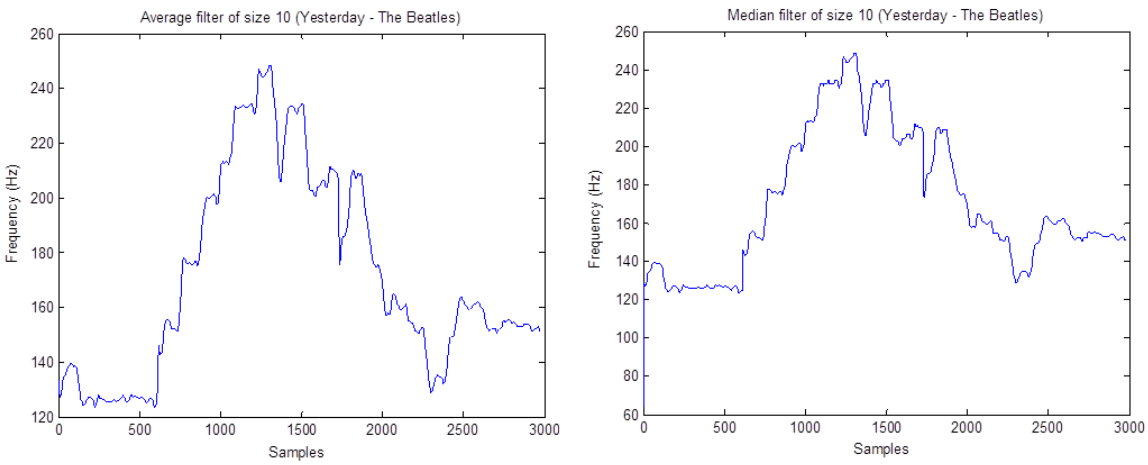

Figure 8: Comparison between average and median filters over the same signal

Once most irregularities have been removed, the following step helps to decrease the computational load of the system, especially in the matching stage. A downsampling function is performed to reduce significantly the number of samples of the signal. Unlike in voice, frequencies (notes) in music are quite stable. Therefore, a sample each 100 miliseconds (with 50 miliseconds of overlapping) is enough to represent the sequence accurately. Thus, 20 frequency samples per second are stored. This step returns a signal with one sample for each seventeen samples in the original signal.

As some irregularities are still present in the signal, it makes no sense to choose just one sample to be taken into account, since an outlier could be picked. Therefore, three downsampling methods involving all the samples have been tested: average, median and mode filters. For each window, one of these filters is applied over all the windowed samples (checking whether it is a voiced or unvoiced frame), and the result is kept as the downsampled signal. When tested under identical conditions, the averaged downsampling has provided the best results.

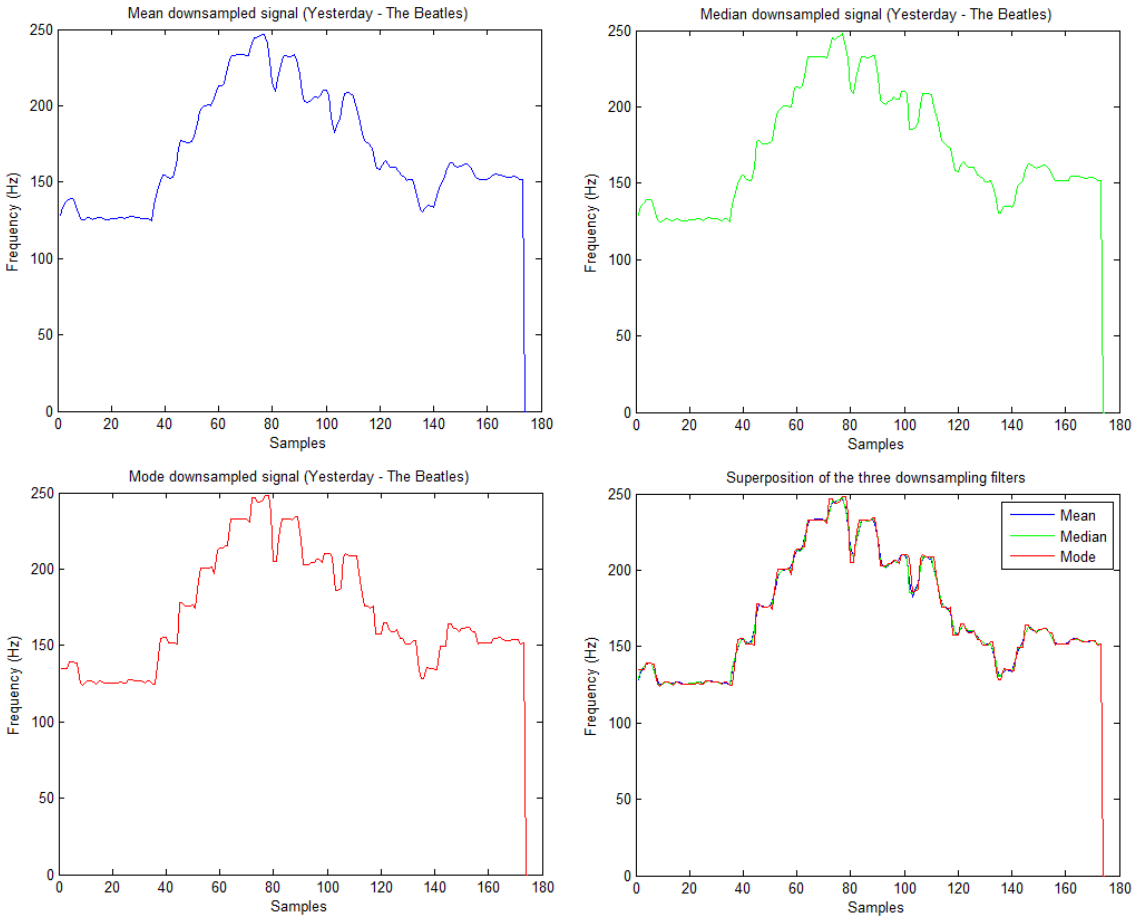

Figure 9: Comparison among downsampling methods

As can be seen in the graphics above (Fig. 9), this step also smoothens the signal.

The next step deals with the scale of the signal: the sequence is mapped from frequency values to MIDI note values. This is a very important step: in the frequency domain, raising an octave means doubling the frequency, which means using a non-linear scale. By converting the frequencies to MIDI values (Fig. 10), the step among notes is normalized, which is important in next steps. The formula to map frequency values into MIDI notes is the following:

$$MIDI = 69 + 12 \cdot log_2(\frac{f}{440})$$

A logarithmic function is used to balance the exponential behavior of the notes represented in the frequency domain.

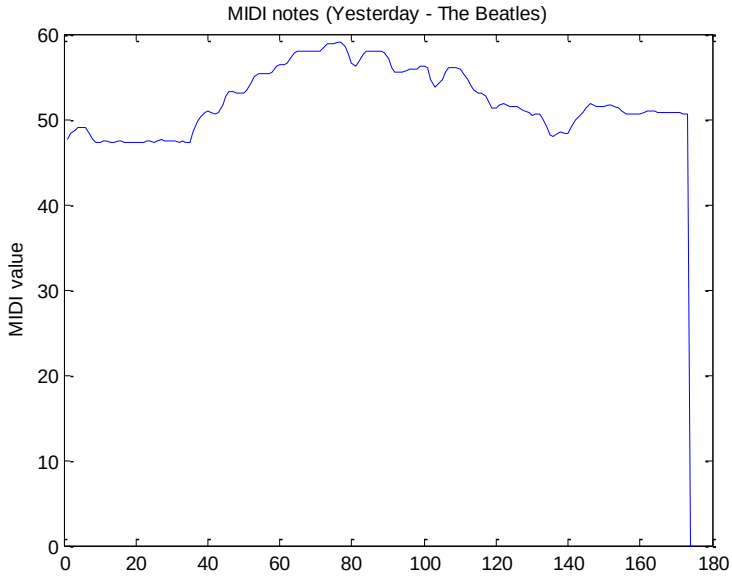

Figure 10: MIDI note representation of the signal

After the mapping, a short median filter is performed to further smoothen the signal.

The final step in the preprocessing stage is, probably, the most important one. If two different users are asked to sing a famous song, they will be very likely to sing in different tonalities, that is, starting the sequence in different notes. And what is more, these notes can also be different to the original note in the recorded song (Fig. 11). One of the most important problems to solve in a QbSH system is how to compare melodies when sung in different tonalities.

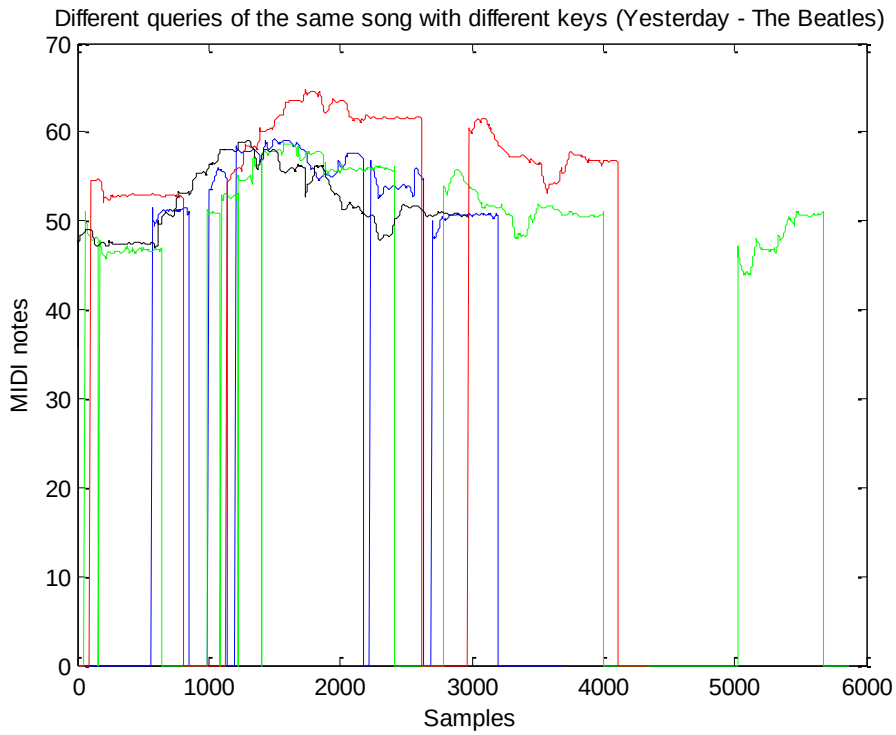

Figure 11: Same extract sung on different keys

A first approach suggested that removing the mean value of a sequence should eliminate key differences, but that would only be true for song segments containing the same musical phrase with good tune and without errors and outliers, and it is known that the system does not accomplish these features. Besides, the pitch sequence of the songs is extracted thoroughly instead of splitting them into phrases.

In this system, the input signal is transformed into a differences signal (Fig. 12). Thus, in a sequence of MIDI notes without silences, a new filter is applied to compute the differences between a sample and the previous one. This allows the signal to be independent of the key in which has been sung. Moreover, this step improves the quality of the signal by correcting some errors. As note differences of more than 12 MIDI notes (an octave) are very uncommon, the difference is computed in modulus 12. Therefore, most octave errors dragged from the extraction process are corrected in this step.

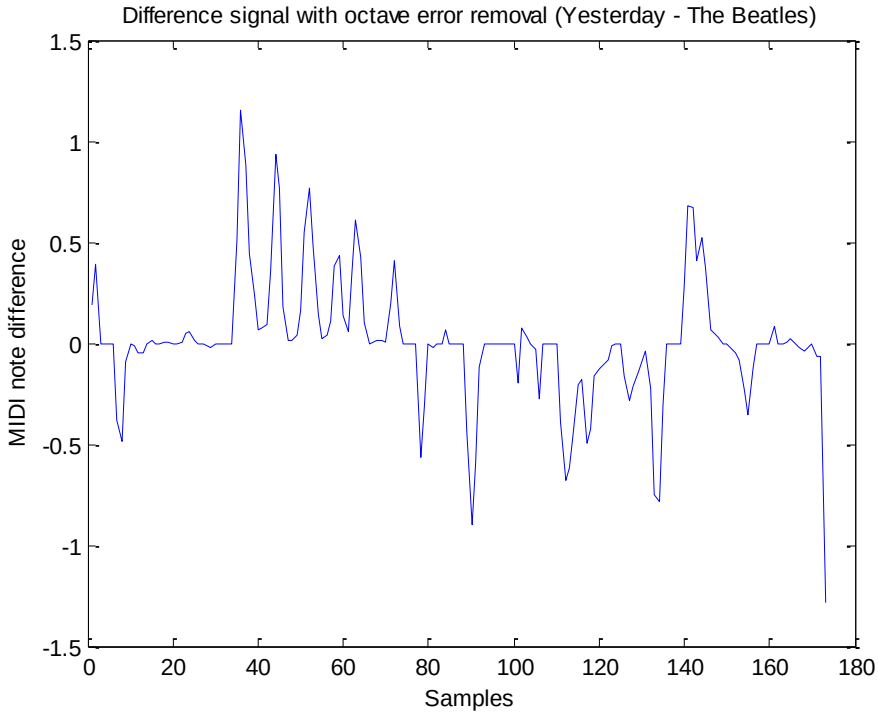

Figure 12: Final signal entering the matching stage

When the system was under development and the option of working with silences was being tested, a similar last step was performed. However, the fact that the difference values in the transitions silence-note and note-silence was different depending on the key had to be taken into account, so a standard penalty was established for these transitions. Nevertheless, this option was finally rejected, as stated before, due to its lower performance.

3.5. Matching: Dynamic Time Warping

The final step in the system is the matching, that is, the comparison among the query and every song in the database to find the closest results. The Dynamic Time Warping (DTW) algorithm is performed to obtain the distances.

The DTW algorithm is very useful for a QbSH system, since it allows a comparison between two signals with time variations (Fig. 13), which is the case of a query sung by two different users: they will probably not sing at the same exact tempo, and that will also happen, as seen before, with the key.

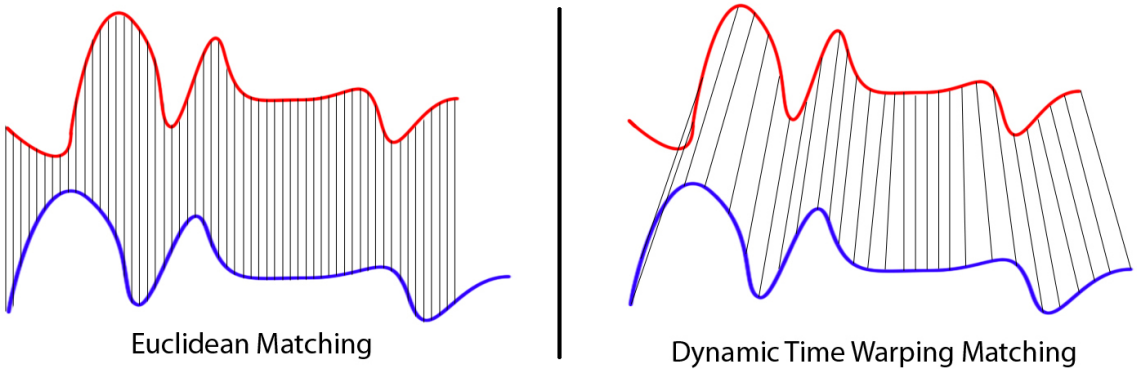

Figure 13: Euclidean Matching vs DTW Matching

The basic idea of the algorithm is computing the path relating the samples of the two signals being compared which has minimum cost. This cost is computed by adding the differences between the values of the two samples being compared in a certain point of the path. Therefore, what is being made is, in the end, some sort of mapping between the two signals being compared. Depending on the application, different path patterns can be allowed.

An interesting point to remark is the fact that the basic DTW algorithms usually try to match the two signals from start to end. In this system, that idea is useless, since trying to match a 10-seconds-length query with a complete song, which can last for several minutes, makes no sense. For example, if a query contains the chorus of a song, the algorithm should allow the matching process to start from the chorus of the song, avoiding the initial part of the song.

Thus, the algorithm has been modified to allow the start and end samples of the query to be matched from any starting point in the song. Therefore, the DTW algorithm is implemented with this formula:

$$\text{DTW} \begin{cases} Initialization: \begin{cases} D(i,j) = 0 \quad if \quad i \in \{0.1\} \\ D(i,j) = -1 \quad otherwise \end{cases} \\ Iteration: D(i,j) = d(x_i, y_j) + \min(S_{path}) \end{cases}$$

Where S_{path} is the set of values of the accumulated cost matrix D belonging to allowed positions in the previous path step, and $d(x_i, y_j)$ is the cost function related to the samples i and j of the matched vectors, x and y, respectively. The basic DTW configuration is shown in the following plot (Fig. 14). For this configuration, for example, and using the previous notation, $S_{path} \in \{ D(i-1,\ j),\ D(i,\ j-1),\ D(i-1,\ j-1) \}$.

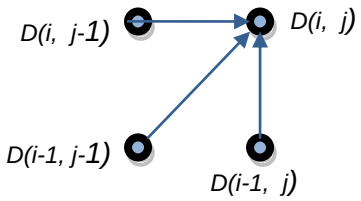

Figure 14: Basic DTW configuration

In order to respect the temporal arrangement of query and templates, the path should be constrained. This is done limiting the values of i and j, as they are forced to increase in each step of the algorithm. This features can also be seen in figure 14. However, this option had an important drawback: it allowed horizontal and vertical paths. As a consequence, unreal mappings showed up, especially when matching pitch sequences with silences (frequency = 0). Figure 15 shows an example of these kind of paths.

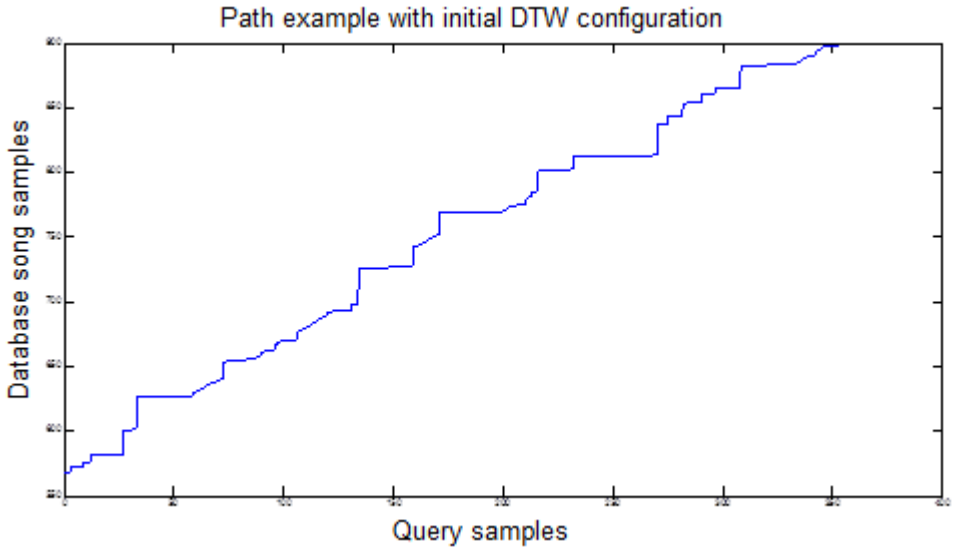

Figure 15: Example of path with basic DTW configuration

As can be seen, there are several vertical and horizontal segments in the path. It makes no sense, since the algorithm is matching a single sample from one of the signals with a lot of samples from the other signal.

In order to fix this, the allowed paths were changed to the ones in the figure 16.

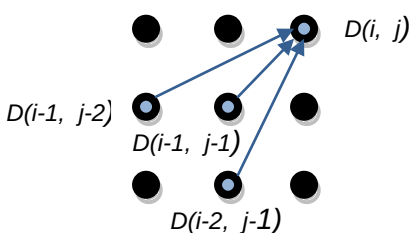

Figure 16: Second DTW path configuration

In this way, having only diagonal paths avoided mapping a single sample with many others. As can be seen, the path was able to grow at double, half, or exactly the same speed amongst the length of the signals. But these paths also had a problem: they could skip samples in columns or rows, that is to say, some of the samples of the signals were ignored if the related cost had been too high. This turned out to be an inadvisable option, since most of times the skipped samples were the peaks of the signals, which contain very important information of the note changes.

Finally, an alternative option has been used (Fig. 17). It allows vertical and horizontal moves within a single sample step, but forces the full path to grow diagonally.

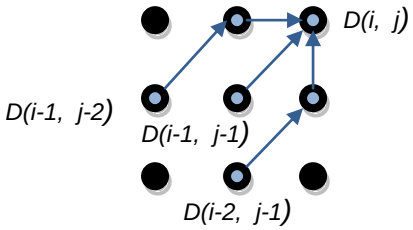

Figure 17: Final DTW path configuration

As this configuration avoids paths from skipping samples and, at the same time, lets paths grow diagonally, it provided the best performance with a noticeable difference. A path (which grows diagonally) and cost function plots can be seen (Fig. 18).

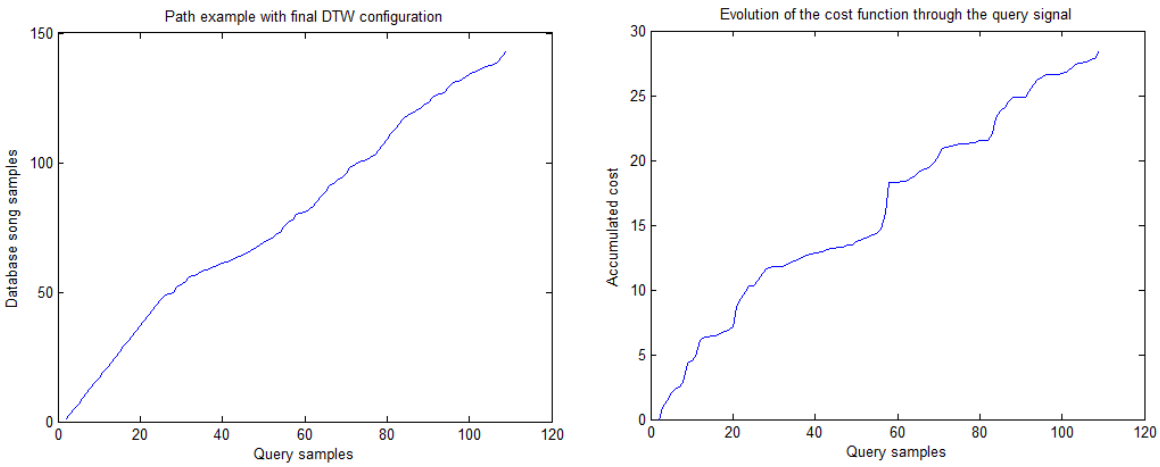

Figure 18: Path and cost evolution from the matching of the same signals

Therefore, using this algorithm with the previous path setting, the matching stage is done, and the baseline system is completed.

From the viewpoint of the computational load, this stage demands the biggest effort. The Matlab programming language is quite intuitive and easy to program, but it is not very efficient. To solve that, a basic DTW implementation written in C code has been used as base code. Then, it has been modified until reaching the final result, and compiled as a C

code inside Matlab environment. This made the execution time between 20 and 30 times faster.

In this chapter, the construction of the baseline system has been reviewed. In the next one, the performance of the system is properly assessed.

4. Results

Once the system is built, several tests can be executed to assess the performance of the complete program. The tests comprise different types of evaluation, including different metrics and different databases.

Firstly, the creation of the databases, which have been used to assess the system, is explained. Next, the metrics for the evaluation of the system are reviewed. Finally, the results of the system tested under different conditions are discussed.

4.1. Database creation

Three different databases have been built to test the system. The first two are created by collecting audio signals, while the third consists in a set of queries obtained from the MIREX[3] contest site.

4.1.1. The Beatles database

The first one, which has been used for most tests, is a 58-song database containing 4 hours of music, formed by the greatest hits from the English band *The Beatles*. This type of music is quite melodic, and the instrumental load is normally not excessive, allowing a reasonably good pitch extraction process. However, *The Beatles* used many vocal harmonies within their songs, which made the extraction process more inaccurate. For this database, 28 queries have been recorded, some of them belonging to the same songs.

4.1.2. The mixed songs database

On the other hand, the second database is a bigger one, more focused on recreating the conditions for the system to be used as a real application. It contains 200 songs, making almost 20 hours of music, mixing a wide range of genres (rock, metal, pop, electronic, latin, etc). In this case, 22 queries were recorded for the testing process. This database, though, has been only used to assess the final performance with the best-case setting.

[3] MIREX 2013: http://www.music-ir.org/mirex/wiki/2013:Query_by_Singing/Humming

4.1.3. Query database

Finally, a third database has been used for a completely different test, the Roger Jang's MIR-QBSH corpus[4]. In this case, the database is also formed by queries. Therefore, this would be a query vs. query test, which would allow for the system not only to be compared against other systems, but also to be tested under the particular context of having a database composed of different reference signals.

The database contains 4431 queries, which are divided in a train database (2391 queries) and a test database (2040 queries). The metrics for this particular test are copied from the MIREX contest, in order for the system to be compared.

4.2. Assessment metrics

The assessment of the system depends on the type of test performed. For a single query, the top-10 list is returned, that is, the 10 results with minimum distance (cost) in the matching process. For the evaluation of a complete set of queries, the Mean Reciprocal Rank (MRR), already reviewed in the second chapter, is computed instead:

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i}$$

4.3. Experimental evaluation in the query vs. audio framework

The following table presents some of the most important results. All the tests containing changes in every particular parameter can be checked in the appendix section at the end of the document.

As can be seen (Table 2), the tests including silences are not as good as the ones in which the silence removal step has been applied. This is especially true for *The Beatles* database, where the MRR increases from 0'19 to 0'45 or from 0'13 to 0'52, depending on the DTW configuration.

Moreover, under these conditions, the third and definitive DTW path mode, which avoided sample skipping, outperforms the second mode, increasing from 0'45 using the second DTW path mode (Fig. 16) to 0'52 in the final DTW path mode (Fig. 17).

The last two rows of the table show the results with the 200 varied songs database. *The Beatles* database obtains a higher MRR (0'52) than the 200 varied songs database (0'25). This is, in fact, totally logical, since the latter database is around 4 times bigger, and the music style of the first one allows for a more accurate melody extraction step.

[4] Roger Jang's MIR-QBSH corpus: http://mirlab.org/dataSet/public/MIR-QBSH-corpus.rar

Parameters	Database	MRR
Silences + Second DTW path mode	The Beatles	0.19
Silence removal + Second DTW path mode	The Beatles	0.45
Silences + Third DTW path mode	The Beatles	0.13
Silence removal + Third DTW path mode	**The Beatles**	**0.52**
Silences + Third DTW path mode	200 varied songs	0.23
Silence removal + Third DTW path mode	**200 varied songs**	**0.25**

Table 2: Representative results of the assessment of the system

The interpretation of the results is reasonably open. It could be said, for the best-performing case with *The Beatles* database, that every song is recognized in the second position of the sorted list in average. Another interpretation could be that half of the songs are perfectly recognized in the first position of the sorted list, while the other half is recognized in the lower positions of the list.

The truth is that the behaviour for each query varies: some of them return the valid result on the first position, some of them somewhere around the middle positions, and others are not correctly identified. The same case could be stated with the 200 varied songs database, but returning the valid result in fourth position in average or recognising perfectly one of each four queries.

It is also important to remark that the assessment of the system is an approximation. The number of queries for each database is low, so the significance of these tests is not very high.

4.4. Experimental evaluation in the query vs. query framework

The alternative query vs. query test has been performed by dividing a query dataset provided by the MIREX contest. The dataset, containing 4431 queries, has been split in testing and training databases.

This test has been assessed with a different metric. For each query, a point is scored for each valid result among the top-10 list. Thus, a query can get a maximum of 10 points on its evaluation and a minimum of 0 points. The average value containing the results of every test query has been computed, being 8’21 the result. Compared to other MIREX[5]

[5] http://nema.lis.illinois.edu/nema_out/mirex2013/results/qbsh/qbsh_task2_jang/summary.html

systems, it is the third best result, being 9'52 the score of the best-performing system (Table 3).

Algorithm	Raw Count
BS1	8.9142
MR1	9.5235
YJ1	7.0034
YJ2	6.8358
Tested system	**8.21**

Table 3: MIREX 2013 QbSH (subtask2) results compared with the tested system

In addition, the MRR has also been computed for this test to ease comparisons with the query vs. audio system. The MRR in this case has been 0'66, which is quite higher than the best query vs. audio case.

This result reflects one important thing: the query vs. audio systems have to improve in many ways to be used as commercial products, especially in the pitch extraction stage, whereas the query vs. query test proved to be the best-performing system.

As for the execution time, the results are fairly good. The evaluation of a normal query (around 10 seconds of length) takes just a few seconds (less than 10 seconds with the largest database), making this feature of the system suitable for future applications. However, for bigger databases, optimization techniques should be analyzed in order to decrease the search time.

5. Graphic User Interface

In this section, a basic review of the accessibility of the system is discussed.

In order to provide the system with higher usability and with analysis tools, a Graphic User Interface (GUI) has been created. The procedure is very simple: a query can be selected or recorded. Once the pitch has been automatically extracted, the query can be played or plotted. Also, a synthetic playback of the extracted pitch can be heard. Finally, the query can be evaluated against a database, which can also be selected.

Once the results have been computed, the top-10 list is shown. Each result stores the part of the song with which the query has been related, and can also be played in both real and synthetic forms, or plotted in the same way as the query.

The plots offer the chance to graphically analyze the evolution of the signals through the preprocessing stage (Fig. 19). Moreover, up to 10 simultaneous results (that is, every result in the top-10 retrieved list) can be plotted simultaneously. Some plotting tools, as zoom and point selection, are also available.

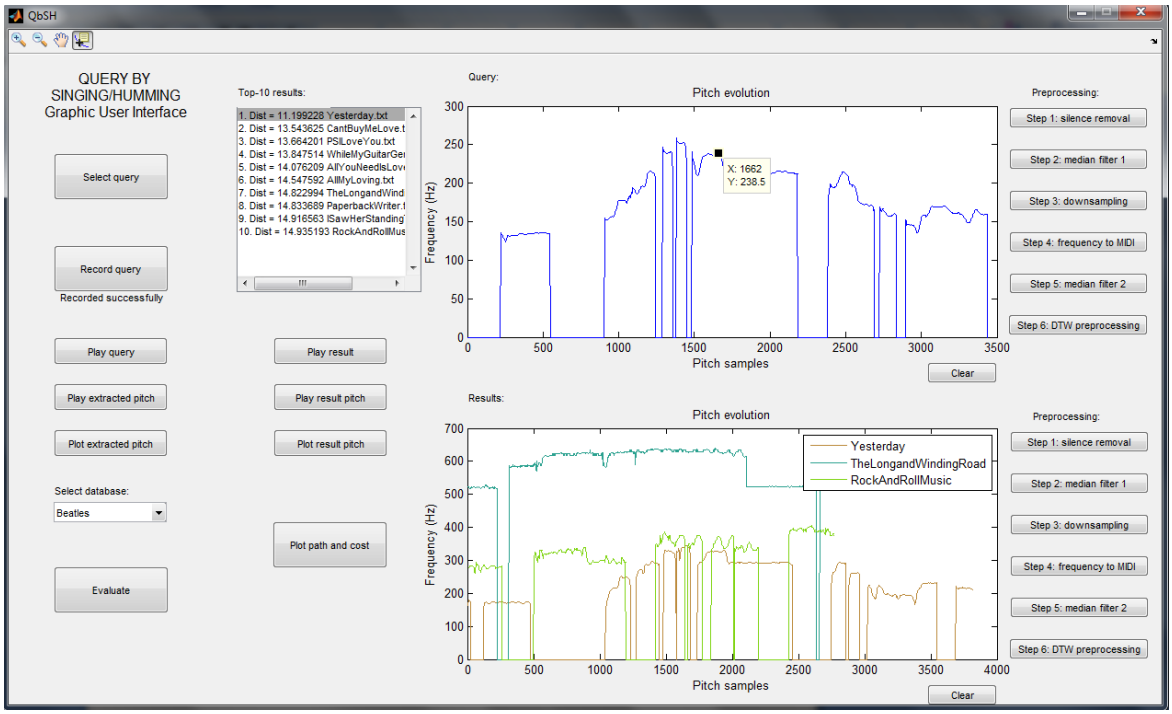

Figure 19: GUI showing plots and preprocessing options

Another type of graphic that can be displayed is the DTW path and cost function for the selected result (Fig. 20). This option can be particularly useful when the user wants to analyze the behaviour of the matching algorithm.

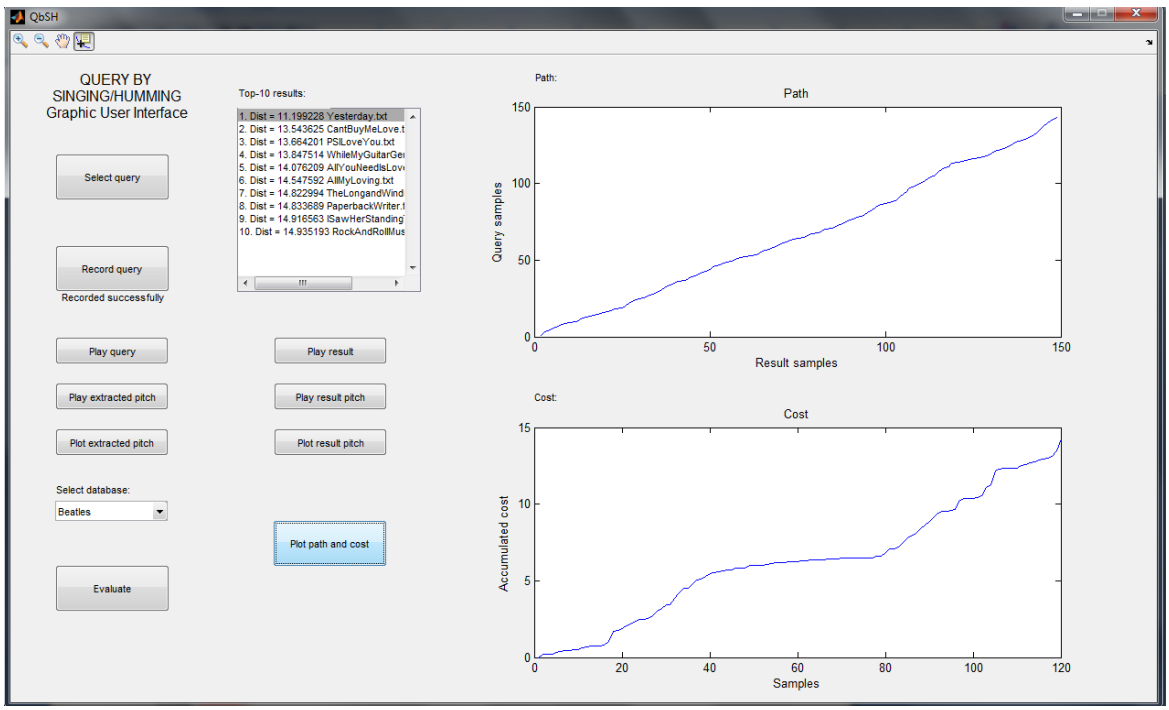

Figure 20: GUI showing path and cost options

6. Budget

This project has been completely developed using a computer and some software, so the main cost to be taken into account is the salary of the project developer and supervisor.

Furthermore, a Matlab license is also required, as the project has been mainly developed under a Matlab environment, and the amortization of the computer has been taken into account as well.

Item	Quantity	Price	Cost
Supervisor (wage)	46 hours	25 €/hour	1150 €
Junior engineer (wage)	690 hours	8 €/hour	5520 €
Matlab (student version)	1 license	124 €/license	124 €
Computer (amortization)	1	5.2 €/week	120 €
		TOTAL	**6914 €**

Table 4: Estimation of the total cost of the project

7. Conclusions and future development:

In this thesis, the main operation of a QbSH system has been analyzed. Different system configurations have been tested and compared.

The system has used audio files as reference signals, although some tests with queries have been made. These files have been processed to build different databases for the program.

A state-of-the-art pitch extraction algorithm for polyphonic signals has been used in the feature extraction step. Many preprocessing techniques have been tried and analyzed to prepare the signal for the matching stage. Based on experimental results, silence removal, downsampling and difference computation steps improved the performance of the system in a very noticeable way.

The well-known Dynamic Time Warping has been used to compute the similarity among the query signal and the songs in the database. Different alternatives have been studied and assessed, being the third one (forcing diagonal paths and avoiding sample skipping) the best-performing option.

Finally, in order to make the system user-friendly, a complete Graphic User Interface has been developed, allowing the user not only to execute the algorithm but to analyze the whole process. This can be useful for further research in the field.

Two audio databases have been collected, one based on *The Beatles* greatest hits, and the other reuniting songs from different music styles. Moreover, a query database has been used for some additional tests.

The obtained results are promising, achieving a MRR of 0’52 in *The Beatles* database and 0’25 in the 200 varied songs database. Furthermore, the third best result is achieved when the query vs. query test in the MIREX contest is run. The MRR in this last case is 0’66, outperforming the query vs. audio test.

The execution time is reasonably low, although there is clearly a lot of room for improvements in every stage of the system. Bearing this in mind, a baseline system has been set up, which is completely functional and allows for changes and enhancements.

Query by Singing/Humming is still a relatively young topic with much research to be done. At the moment, query vs. audio might be the worst-performing type of system but, in the future, it could allow for the automation of the whole process, since audio files do not require any kind of human work either in the obtaining of the signal or in the feature extraction.

Bibliography

Antonelli, M., Rizzi, A., & del Vescovo, G. (2010). A Query by Humming System for Music Information Retrieval. *10th International Conference on Intelligent Systems Design and Applications.*

Cannam, C., O. Jewell, M., Rhodes, C., Sandler, M., & d'Inverno, M. (2010). Linked Data and You: Bringing music research software into the Semantic Web. *Journal of New Music Research, volume 39, no. 4* , pp. 313-325.

Dong, Y., & Qi, B. (2010). The Technology of Music Retrieval by Humming and Its Application in Internet Music Search System. *IEEE* .

Guo, Z., Wang, Q., Liu, G., & Guo, J. (2013). A query by humming system based on locality sensitive hashing indexes. *Signal Processing 93* , 2229-2243.

Guo, Z., Wang, Q., Yin, L., Liu, G., & Guo, J. (2012). Query by Humming via Hierarchical Filters. *21st International Conference on Pattern Recognition.*

Ito, A., Kosugi, Y., Makino, S., & Ito, M. (2010). A Query-by-Humming Music Information Retrieval from Audio Signals based on Multiple F0 Candidates. *IEEE International Conference on Audio, Language and Image Processing.*

Jain, P. K., Jain, R., Patil, H. A., & Basu, T. K. (2011). *International Conference on Asian Language Processing.*

Jang, D., Song, C.-J., Shin, S., Park, S.-J., Jang, S.-J., & Lee, S.-P. (2011). Implementation Of A Matching Engine For A Practical Query-by-Singing/Humming System. *IEEE* .

Kotsifakos, A., Papapetrou, P., Hollmén, J., Gunopulos, D., Athitsos, V., & Kollios, G. (2012). Hum-a-song: A Subsequence Matching with Gaps-Range-Tolerances Query-By-Humming System. *38th International Conference on Very Large Data Bases.*

Li, J., Han, J., Shi, Z., & Li, J. (2010). An Efficient Approach to Humming Transcription for Query-by-Humming System. *3rd International Congress on Image and Signal Processing.*

Midomi - SoundHound. (s.f.). Recuperado el 10 de July de 2014, de http://www.midomi.com/

Music Technology Group - Universitat Pompeu Fabra. (s.f.). *MELODIA vamp plug-in*. Obtenido de MTG - UPF: http://mtg.upf.edu/technologies/melodia

Phiwma, N., & Sanguansat, P. (2010). A Novel method for Query-by-Humming Using Distance Space. *First International Conference on Pervasive Computing, Signal Processing and Applications.*

Rocamora, M., Cancela, P., & Pardo, Á. (2013). Query by humming: Automatically building the database from music recordings. *Pattern Recognition Letters 36* , 272-280.

Salamon, J., & Gómez, E. (2012, August). Melody Extraction From Polyphonic Music Signals Using Pitch Contour Characteristics. *IEEE Transactions on audio, speech, and language processing, vol. 20, no. 6* .

Salamon, J., Gómez, E., Ellis, D. P., & Richard, G. (2014). Melody Extraction from Polyphonic Music Signals: Approaches, Applications and Challenges. *IEEE Signal Processing Magazine* .

Salamon, J., Serrà, J., & Gómez, E. (2013). Tonal representations for music retrieval: from version identification to query-by-humming. *Int. J. Multimed. Info. Retr.*

Song, C.-J., Park, H., Yang, C.-M., Jang, S.-J., & Lee, S.-P. (2013). Implementation of a Practical Query-by-Singing/Humming (QbSH) System and Its Commercial Applications. *IEEE Transactions on Consumer Electronics, vol. 59, no. 2* .

Talkin, D. (1995). A Robust Algorithm for Pitch Tracking (RAPT). En *Ch. 14 in Speech Coding and Synthesis.*

Tsai, W.-H., & Tu, Y.-M. (2012). An Efficient Query-by-Singing/Humming System Based on Fast Fourier Transforms of Note Sequences. *IEEE International Conference on Multimedia and Expo.*

Tsai, W.-H., Tu, Y.-M., & Ma, C.-H. (2012). An FFT-based fast melody comparison method for query-by-singing/humming systems. *Pattern Recognition Letters 33* , 2285-2291.

Appendix I: full results

This appendix contains the results from most performed tests. Most of them refer to modifications on the baseline system with the baseline preprocessing: first median filter, average filter, silence detection and interpolation, downsampling, MIDI mapping and second median filter. Note that (dtw2) and (dtw3) make reference to the second and third DTW path options, respectively.

Parameters	Database	MRR
Differences (dtw2)	The Beatles	0.1456
Differences + silence removal (dtw2)	The Beatles	0.2109
Mean subtraction (dtw2)	The Beatles	0.1380
Mean subtraction + silence removal (dtw2)	The Beatles	0.2132
Differences (dtw3)	The Beatles	0.0652
Differences + silence removal (dtw3)	The Beatles	0.2035
Differences + no median filter (dtw2)	The Beatles	0.0927
Differences + no median filter (dtw3)	The Beatles	0.11
Differences + silence removal + no median filter (dtw2)	The Beatles	0.1012
Differences + silence removal + no median filter (dtw3)	The Beatles	0.0959
Differences + no interpolation (dtw2)	The Beatles	0.1541
Differences + no interpolation (dtw3)	The Beatles	0.0891
Differences + silence removal + no interpolation (dtw2)	The Beatles	0.0841
Differences + silence removal + no interpolation (dtw3)	The Beatles	0.0974

Differences + no average filter (dtw2)	The Beatles	0.1949
Differences + no average filter (dtw3)	The Beatles	0.1329
Differences + silence removal + no average filter (dtw2)	The Beatles	0.1692
Differences + silence removal + no average filter (dtw3)	The Beatles	0.3160
Differences + median filter start (50) + median filter end (10) (dtw2)	The Beatles	0.19
Differences + median filter start (50) + median filter end (10) (dtw3)	The Beatles	0.1307
Differences + silence removal + median filter start (50) + median filter end (10) (dtw2)	The Beatles	0.1961
Differences + silence removal + median filter start (50) + median filter end (10) (dtw3)	The Beatles	0.3086
Differences + median + average (dtw2)	The Beatles	0.1052
Differences + median + average (dtw3)	The Beatles	0.1003
Differences + silence removal + median + average (dtw2)	The Beatles	0.2269
Differences + silence removal + median + average (dtw3)	The Beatles	0.2533
Baseline + single octave mapping (dtw2)	The Beatles	0.1376
Baseline + single octave mapping (dtw3)	The Beatles	0.1268

Table 5: Additional results (I)

Now, the following results are computed over the same baseline system but removing the silence in every case just after the pitch detection instead of after the smoothing filters.

Parameters	**Database**	**MRR**
Differences (dtw2)	The Beatles	0.4496
Differences (dtw3)	The Beatles	0.4951
Differences + no interpolation (dtw2)	The Beatles	0.4493
Differences + no interpolation (dtw3)	The Beatles	0.4950
Differences + double median (dtw2)	The Beatles	0.4452
Differences + double median (dtw3)	The Beatles	0.5177
Differences + average + median (dtw2)	The Beatles	0.4445
Differences + average + median (dtw3)	The Beatles	0.5128
Differences + no median (dtw2)	The Beatles	0.2937
Differences + no median (dtw3)	The Beatles	0.2859
Differences + average instead of second median (dtw2)	The Beatles	0.3820
Differences + average instead of second median (dtw3)	The Beatles	0.3799
Differences + window size change in filters (10 to 5) (dtw2)	The Beatles	0.3218
Differences + window size change in filters (10 to 5) (dtw3)	The Beatles	0.3598
Optimizing filter sizes: avg10 to avg8 (dtw3)	The Beatles	0.5161
Optimizing filter sizes: avg10 to avg6 (dtw3)	The Beatles	0.5161

Optimizing filter sizes: avg10 to med6 (dtw3)	The Beatles	0.5179
Optimizing filter sizes: avg10 to med6 + dwnsmplmean to dwnsmplmode (dtw3)	The Beatles	0.3648
Best case with silence	200 varied songs	0.2281
Best case + silence removal	200 varied songs	0.2521

Table 6: Additional results (II)

Peak integration tests, made to avoid note changes during several samples. Tests applied over the best-performing case:

Parameters	**Database**	**MRR**
Peakint (th=0.1)	The Beatles	0.3048
Peakint (th=0.1) + peakcl	The Beatles	0.2966
Peakint (th=0.2)	The Beatles	0.2561
Peakint (th=0.05)	The Beatles	0.2249
Peakint (th=0.001)	The Beatles	0.2713

Table 7: Additional results (III)

Comparison among different downsampling methods:

Parameters	**Database**	**MRR**
Mean downsample (dtw3)	The Beatles	0.5139
Mean downsample (dtw2)	The Beatles	0.444
Mode downsample (dtw3)	The Beatles	0.3532
Mode downsample (dtw2)	The Beatles	0.2809
Median downsample (dtw3)	The Beatles	0.3603
Median downsample (dtw2)	The Beatles	0.3246
Silence removal + median downsample (dtw3)	The Beatles	0.3807

Silence removal + median downsample (dtw2)	The Beatles	0.302

Table 8: Additional results (IV)

Comparison among different window lengths on the downsampling process:

Parameters	**Database**	**MRR**
Window 0.06 seconds	The Beatles	0.4
Window 0.09 seconds	The Beatles	0.4248
Window 0.09 seconds	The Beatles	0.5139
Window 0.11 seconds	The Beatles	0.4635
Window 0.15 seconds	The Beatles	0.4808
Window 0.2 seconds	The Beatles	0.4004

Table 9: Additional results (V)

Best case with RAPT for query pitch extraction:

Parameters	**Database**	**MRR**
Best case + RAPT (query)	The Beatles	0.39
Best case + RAPT (query)	200 varied songs	0.09

Table 10: Additional results (VI)

Query vs. query results, with best case configuration:

Test	**Result**
MRR	0.6563
Average number of correct queries in first 10 results	8.2
Percentage of queries retrieving a correct song in first 10 results	90.33%

Table 11: Additional results (VII)

Appendix II: code download

The Matlab code of the system is available for download[6]. The RAR file contains the basic folder structure with Matlab *.m* files.

Note that the databases are not included and that paths and folders should be adjusted in order for the code to work. Recommended only to check out the functions individually.

[6] Matlab code: http://goo.gl/J26pgC

Glossary

QbSH: Query by Singing/Humming

MIDI: Music Instrument Digital Interface

DTW: Dynamic Time Warping

MRR: Mean Reciprocal Rank

GUI: Graphic User Interface

FFT: Fast Fourier Transform

SVM: Support Vector Machines

ANN: Artificial Neural Networks

RAPT: Robust Algorithm for Pitch Tracking

MIREX: Music Information Retrieval Evaluation eXchange