UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

QUERY BY HUMMING

(Android App)

Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Marc Siquier Penyafort

In partial fulfilment

of the requirements for the degree in

AUDIOVISUAL SYSTEMS ENGINEERING

Advisor: Antonio Bonafonte Cávez

Barcelona, February 2015

Abstract

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

It has been developed as a client/server system, where the client is an Android application (programmed on Java) and the server is located on a Unix system and written on C++.

The system compares a melody recorded by the user with other melodies previously recorded by other users and tagged with song information by the system administrator.

A pitch extraction algorithm is applied in order to extract the melody for the query songs, then a processing algorithm in order to enhance the signal and prepare it for the matching. In the matching step Dynamic Time Warping (DTW) has been applied, which computes a distance between two signals and absorbs tempo variations.

As a result, this thesis contains a full experience of audio processing, systems administration, communications and programming skills.

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.

Ha estat desenvolupat com un sistema client/servidor, on el client és una aplicació Android (programada en Java) i el servidor està basat en una màquina Unix i escrit en C++.

El sistema compara una melodia gravada per l'usuari amb altres melodies prèviament gravades per altres usuaris i etiquetades amb informació de la cançó pel propi administrador del sistema.

Per a extreure la melodia dels fragments gravats per l'usuari, s'ha aplicat un algoritme d'extracció de *pitch*. Posteriorment s'ha aplicat un preprocessat per a millorar la senyal i preparar-la per a la classificació. A l'etapa de classificació s'ha aplicat el *Dynamic time Warping* (DTW), que calcula la distància entre dues senyals absorbint variacions temporals.

Així, aquesta tesi conté una experiència completa en processat d'àudio, administració de sistemes, comunicacions i habilitats en programació.

Resumen

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

El sistema ha sido desarrollado como un sistema cliente/servidor, donde el cliente es una aplicación Android (programada en Java) y el servidor está basado en una máquina Unix y escrito en C++.

El sistema compara una melodía grabada por el usuario con otras melodías previamente grabadas por otros usuarios y etiquetadas con información de la canción por el propio administrador del sistema.

Para extraer la melodía de los fragmentos grabados por el usuario, se ha aplicado un algoritmo de extracción de *pitch.* Posteriormente se ha aplicado un preprocesado para mejorar la señal y prepararla para la clasificación. En la etapa de clasificación se ha aplicado el *Dynamic Tiime Warping* (DTW), que calcula la distancia entre dos señales absorbiendo variaciones temporales.

De esta forma, esta tesis contiene una experiencia completa en procesado de audio, administración de sistemas, comunicaciones y habilidades en programación.

Acknowledgements

First of all I want to thank the supervisor of this project Antonio Bonafonte, whose advise and ideas have been fundamentals to this project.

Secondly thanks go to Pau Tur, whose thesis inspired and gave the basis for this project.

Last but not least, I am very grateful to my family and friends for their unconditional support and encouragement over these years. Many thanks.

Revision history and approval record

Revision	Date	Purpose
0	02/01/2015	Document creation
1	06/01/2015	Document revision
2	14/01/2015	Document revision
3	20/01/2015	Document revision
4	27/01/2015	Document revision
5	04/02/2015	Final document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Marc Siquier Penyafort	marcsiquierpenyafort@gmail.com
Antonio Bonafonte Cávez	antonio.bonafonte@upc.edu

Written by:		Reviewed and approved by:	
Date	04/02/2015	Date	04/02/2015
Name	Marc Siquier/ Penyafort/	Name	Antonio Bonafonte Cávez
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables

Glossary

AMDF: Average Magnitude Difference Function

DTW: Dynamic Time Warping

f0: Fundamental frequency

HMM: Hidden Markov models

IIR: Infinite Impulsional response

MDCT: Modified Discrete Cosine Transform

MIDI: Musical Instrument Digital Interface

MIR: Music Information Retrieval

MRR: Mean Reciprocal Range

NCCF: Normalized Cross-correlation function

QbSH: Query byes Singing/Humming

RAPT: Robust Algorithm for pitch tracking

SDK: Software development kit

ZCR: Zero-crossing rate

1. Introduction

Query by humming (QbH) is a system that involves taking a user-hummed melody (input query) and comparing it to an existing database. The system then returns a ranked list of music closest to the input query.

These systems are not very common nowadays, but such a system would be very useful in any multimedia database containing musical data by providing an alternative and natural way of querying. This is because a natural way of querying an audio database (of songs) is to hum the tune of a song.

In this project the wide problem of building a complete functional Query by Singing/Humming system has been dealt with. In the following sections a thorough planning of the whole work is detailed.

1.1. Statement of purpose

The project has been carried out at the Departament de Teoria del Senyal i Comunicacions (TSC) at the Universitat Politècnica de Catalunya (UPC) in Barcelona, Spain, for over 5 months.

The purpose of this project is to experiment with different techniques in the field of the Music Information Retrieval (MIR) and to program a complete functional Android application of a Query by Humming system. The system compares human queries with other human queries saved in a database.

The main goals of the project 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 android app.
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 for further research.
6. Experimenting with some Android language features.
7. Programming a server and its connection to multiple devices.

1.2. Requirements and specifications

System requirements:

- The system 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 behaviour of the whole system should be good, in general terms.
- The App should run normally and smoothly.
- The App should have a user friendly interface.
- The system should give room for innovation and further improvements.
- The project should provide a developed framework for future research.

System specifications:

- The MRR (Mean Reciprocal Rate) should be in the range of the state-of-the-art methods (50%).
- 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 1000).

1.3. Methods and procedures

This project is the continuation of a previous project called "Query by Humming" by Pau Tur Vallés but giving a twist to the problem, because this thesis develops the problem from its beginning and in a different way.

Pau Tur's project was developed as a prototype in order to be useful for future research by providing a baseline system and a Matlab program as tool to implement the system. This project sees the problem from a more "realistic" point of view, integrating a server with a database and an Android app in order to implement the whole system as a client/server.

1.4. Workplan and Gantt Diagram

Work Packages:

<table>
<tr><td>Project: Query By Humming (Android App)</td><td colspan="2">WP ref: 1</td></tr>
<tr><td>Major constituent: Study of the topic</td><td colspan="2"></td></tr>
<tr><td rowspan="2">Short description:
Initial study of the main state-of-the-art techniques applied in the Query by Humming tasks (pitch extracting, filtering, matching,...).
Reading and understanding the Pau Tur Valles project.</td><td colspan="2">Planned start date:01/09/2014
Planned end date:30/09/2014</td></tr>
<tr><td colspan="2">Start event:T1
End event:T3</td></tr>
<tr><td>Internal task T1: Reading Pau's project
Internal task T2: Understanding the implementation of that system.
Internal task T3: Studying the main state-of-the-art techniques applied.</td><td>Deliverables:
Write the final report points 1 & 2</td><td>State:</td></tr>
</table>

<table>
<tr><td>Project: Query By Humming (Android App)</td><td>WP ref: 2</td></tr>
<tr><td>Major constituent: Communications</td><td></td></tr>
<tr><td>Short description:
Study, understand and implement the communications that will</td><td>Planned start date: 17/09/2014
Planned end date: 30/09/2014</td></tr>
</table>

be used to connect server-app and send audio stream and retrieval information. Choose and implement the protocol to be used to communicate server with android mobile.	Start event:T1 End event: T2	
Internal task T1: Program the server in order to receive, process and store an audio stream from the Android App. Internal task T2: Implement the client part of the communications, how the app should send the data, and receive it.	Deliverables: A written report about how the communications will work	State:

Project: Query By Humming (Android App)	WP ref: 3	
Major constituent: Database Creation		
Short description: Record and create the starting database. Study the different kinds of database and implement one to store the metadata and the files itself. Implement a database that will grow up with the users queries that will be automatically added to the server's database.	Planned start date: 01/10/2014 Planned end date: 20/12/2014 Start event: T1 End event: T2	
Internal task T1: Record a few queries for each song and create the dynamic database. Internal task T2: Choose and implement how the new queries will be added to the server database.	Deliverables: A database with about 300 songs to work with.	State:

Project: Query By Humming (Android App)	WP ref: 4	
Major constituent: Technologies to be used		
Short description: Study, choose and implement the technologies that will be used in order to process the audio stream, extracting the pitch, filtering it... Choose the way that the matching will be done. And implement a baseline system in order to improve it later.	Planned start date: 01/10/2014 Planned end date: 30/12/2014 Start event: T1 End event: T2	
Internal task T1: Technologies in order to process the audio Internal task T2: Technologies in order to match the query with the database. Internal task T3: Deliver a base system.	Deliverables: A working base system in order to be improved.	State:

Project: Query By Humming (Android App)	WP ref: 5
Major constituent: Server Implementation	
Short description: Program the server in order to process the input audio stream,	Planned start date: 20/10/2014 Planned end date: 31/12/2014

search in the database and send back to the app the result of the matching process.	Start event:T1 End event:T1	
Internal task T1: Program the server in order to receive, process store, match an audio stream from the Android App.	Deliverables: Installation of the host	State:

Project: Query By Humming (Android App)	WP ref: 6	
Major constituent: Android app programming		
Short description: Program the client (Android App) in order to record a piece of hummed song, get the results from the server and show them to the user.	Planned start date: 20/10/2014 Planned end date: 15/01/2015	
	Start event:T1 End event:T2	
Internal task T1: Design how the app should work, classes and methods to implement, activities...	Deliverables: The app itself	State:

Project: Query By Humming (Android App)	WP ref: 6	
Major constituent: Visual Improvement		
Short description: Re-program the Android app in order to make it visual friendly, changing the colours, background, buttons, ...	Planned start date: 15/01/2015 Planned end date: 20/01/2015	
	Start event:T1 End event:T2	
Internal task T1: Design visually the App. Internal task T2: Change background of Android Activities, colors, buttons, to make the app looks as the design.	Deliverables: The app visually improved.	State:

Project: Query By Humming (Android App)	WP ref: 8	
Major constituent: Report writing		
Short description: Write the report of the project and prepare the oral defence.	Planned start date: 01/01/2015 Planned end date: 30/01/2015	
	Start event:T1 End event:T2	
Internal task T1: Write the final report.	Deliverables: Project report	State:

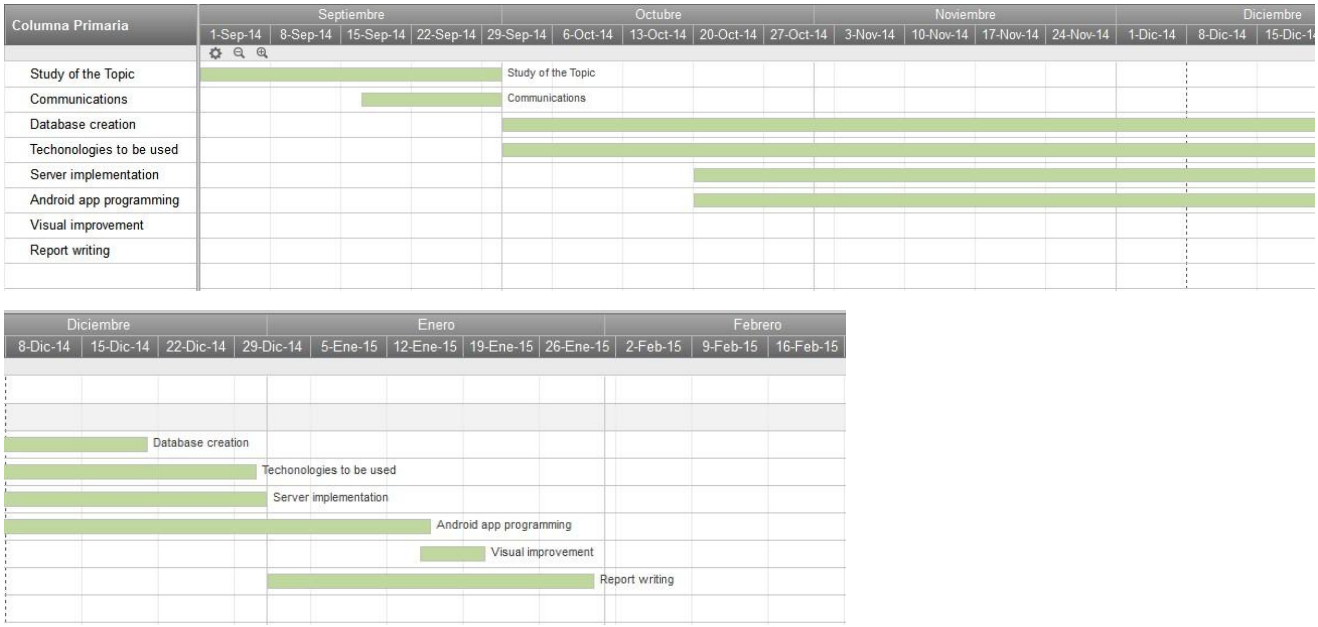


Figure 1: Gantt diagram

1.5. Deviations from the initial plan

I had some problems with the creation of the first database, because the idea was to record extracts from the song most listened to in the country. It was difficult to find this list, because there are no official charts of the songs most listened to, but finally we were able to find one.

The idea was to give the results and compare them with those obtained by Pau Tur in his work. The results are very different because he was comparing audio files with hummed recorded queries and this system is developed and focused on the query vs. query comparison

2. State of the art of the technology used or applied in this thesis:

The general idea of a QbSH system 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 chapter a general overview of all the different methods applied is given.

2.1. Query by Singing/Humming

A QbSH is a relatively recent topic with no universally accepted solution, as proposed solutions vary drastically. In this subchapter some commercial systems and some investigation groups are reviewed.

SoundHound and *Midomi* are the only commercially available query by humming systems available online. Both are powered by the same backend and are capable of recognizing humming and singing as well as recorded tracks. *SoundHound* is the mobile app version of the web service *Midomi*.

SoundHound and *Midomi* are powered by their own *Sound2Sound Search Science,* and ant the moment of writing this thesis, the algorithm it uses has not been made public.

For the singing and humming search, the searchable database is based on Midomi.com users's contributions (global community with over 230 million users), making it the largest database of its kind by a large margin. *Midomi*'s singing/humming search is a system based on the query vs. query matching, and also has a speech recognition system that tries to match the lyrics.

Musipedia is another example of a QbSH system that uses a variety of input methods such as humming, tapping the keyboard, keyboard search (a virtual piano keyboard), draw notes, contour search, and also with parsons code[1]. *Musipedia* uses its own *Melodyhound* melody search engine.

There are a few small commercial systems such as *QueryHammer* based in MATLAB and using the features described on the MPEG-7 standard. *Sloud*, which is an ActiveX applet, is based on diffuse search algorithms, which allows errors and imprecision on the query. Finally, there is *A Tarter 2003*, an applet using 100000 midi files to match the queries directly recorded from de web browser.

Despite these commercial systems, there are several groups researching for this topic. The Working Group For Electronic Media Technology (AEMT) from the Fraunhofer institute in Ilmenau, Germany; works in different MIR fields, and one of them is a QbSH system based in the AudioID project and using the MPEG-7 standard.

The College of Information Science and Technology (IST) in Penn State, Pennsylvania, is working in QbSH projects oriented to MP3, which are harder to handle. "A New Spectral Based Approach To Query By Humming For MP3 Songs Database" its based in spectral processing, MDCT (Modified Discrete Cosine Transform) coefficients analysis and peak energy detection in order to ignore background music and only process the melody.

[1] Each pair of consecutive notes is coded as "U" (up) if the second note is higher than the first note, "R" (repeat) if the pitches are equal and "D" (down) otherwise.

In Cornell University, in New York they are also researching and developing QbSH systems based in pitch tracking algorithms and MIDI databases.

Charles Parker, from Oregon State University, developed a method that uses artificial intelligence techniques in order to solve the computational load problem of the QbSH algorithms. See his work "Applications of Binary Classification and Adaptive Boosting to the Query by Humming Problem".

2.2. Reference signals

The first choice of a QbSH system is what kind of signals will form the database. Namely audio files, MIDI files or queries can be used to form that database.

Audio files are the most common and accessible way to store music and there are many different audio file extensions depending on the compression, the quality, etc. Using this kind of files the database can grow easier and faster, unlike other kind of signals. However, extracting features from these files is harder than from the others. This is because the features are directly extracted from the raw mix of musical instruments that can be in every typical song.

MIDI (Musical Instrument Digital Interface) is a standard protocol of music description and communication which is widely used by electronic musical instruments and computers. Inside a MIDI file, information of the different musical instruments is stored in separated tracks, and each feature of a note (value, duration, tone, effect, etc.) is represented by numbers. Therefore, the main melody is contained in just one track; the goal is to identify this track if it is not tagged and also extract the melody from it.

Finally, short recordings sung by different people can be used to form the database. This method gives the best matching results because queries are compared between them and the variety of singers and queries of the same song can strengthen the reliability of the database.

However, there is a big problem in order to start the database because a good database requires plenty of recordings. The idea of the database is to grow as the system is used because the files are obtained from the real queries of the users.

2.3. Pitch Detection

The pitch (or musical pitch) of an audio signal is a perceptual feature, relevant only in the context of a human listening to that signal. Pitch is loosely related to the log of the frequency, perceived pitch increasing about an octave with every doubling in frequency.

The perception of pitch changes with this harmonic content as well. A richer spectrum seems to reinforce the sensation of pitch, making the sound seem more "in-tune".

The closest feature to pitch is the Fundamental frequency (f0). Fundamental frequency estimation, also referred to as pitch detection, is defined as the detection of the greatest common divisor of the harmonics in a particular segment of a signal, that is, in its spectrum.

As all the audio signals are very different (depending on the instruments, the singer, etc.) it has been difficult to develop a "context-free" f0 estimator. The result is that there are many f0 estimators currently on the market, but few are appropriate for more than one domain. There are three different ways to estimate the f0: in the time domain, in the frequency domain and in the statistical frequency domain.

2.3.1. Time Domain Methods

The first and most common approach to the f0 estimation problem is to look directly at the waveform that represents the change in air pressure over time and attempt to detect the f0 from that waveform.

There are several methods based on the theory that if a waveform is periodic, then there are extractable time-repeating events that can be counted, and are inversely related to the frequency.

A possible way to count events is to count the number of positive peaks per second in the waveform. The distance between the local maxima gives the wavelength which is inversely proportional to the frequency.

A lot of f0 estimators work with the autocorrelation function of the waveform, using the fact that in periodic waveforms the autocorrelation function itself is also periodic. The first peak in the autocorrelation indicates the period of the waveform.

There are many other ways to estimate the pitch in the time domain like searching for features in the phase space representation of the signal, the yin-estimator, etc.

2.3.2. Frequency Domain Methods

Another way to estimate the fundamental frequency is to look at the frequency representation of the signal. Many attempts have been made to extract and follow the f0 of a signal in this manner.

The first methods were based on the component frequency ratios, trying to identify the fundamental frequency looking on peaks in the spectral transformation of the signals.

Another kind of algorithms are called the Filter-based methods that try to estimate f0 by applying different filters with different centre frequency, and comparing their output. When a spectral peak lines up with the pass-band of a filter, the result is a higher value than when the pass-band does not line up.

One approach are the comb filters f0 estimators. A comb filter has many equally spaced pass-bands so in this kind of algorithms the output will be greatest when the pass-bands of the comb line up with all the harmonics.

A more recent filter-based f0 estimator is the Tuneable IIR filter, which consists of a narrow user-tuneable band-pass filter, which is swept across the frequency spectrum.

Cepstral analysis is a form of spectral analysis where the output is the Fourier transform of the log of the magnitude spectrum of the input waveform. The theory behind this relies on the fact that when the log magnitude of a spectrum is taken, the peaks are reduced, the result is a periodic waveform in the frequency domain, the period of which is related to the fundamental frequency of the original waveform.

2.3.3. Statistical Frequency Domain Methods

In these methods each input frame is classified into one of a number of groups, representing the f0 estimator of the signal. Two main methods are used to estimate the pitch using statistical methods.

The first is to train a neural network with a set of spectral partials, or the time-domain waveform, or the phase space representation of the signal. It is likely to output a frequency hypothesis, which could then be translated to pitch.

There is also a series of papers on f0 estimation using maximum likelihood estimators. For a set of candidate fundamental frequencies, the algorithm computes the probability that a given observation was generated from each f0 in the set, and finds the maximum. The choice of the set of fundamental frequencies is important, because, the observation could originate from any f0.

2.4. Matching

The matching step consists of deciding to which song on the database a query recorded by the user belongs to. That is to give a score to each song in the database that represents the similarity between this song and the query. So, the song with minimum score is the most similar song to the query.

With some research we can observe that most QbSH systems these days use Dynamic Time Warping as a classifier, but there are also different ways to do the matching step.

2.4.1. Dynamic Time Warping

Dynamic time warping is widely extended in almost every QbSH system. DTW is a technique for aligning time series, with its main application being in the domain of speech recognition, for three decades.

Although as a speech recognition technique it has been largely superseded by Hidden Markov Models (HMM), it remains very useful today, more so for its usefulness and flexibility as similarity measure. Moreover, the DTW algorithm can absorb variations in time or speed in the matching step between two songs and that is a very important fact in a QbSH system because time and speed are the most typical imperfections in humans trying to sing a song.

2.4.2. Hidden Markov Models

A Hidden Markov Model (HMM) is a statistical Markov model in which the system being modelled is assumed to be a Markov process with unobserved (hidden) states. A HMM can be considered a generalization of a mixture model where the hidden variables, which control the mixture component to be selected for each observation, are related through a Markov process rather than being independent of each other.

In a QbSH HMM-based approach (as defined by Yi Ma of Ohio State University), a separate HMM is trained for each target song in the database. A query is represented as an observation for the HMMs. The probability of a query given a particular HMM indicates the degree of similarity between the query and target song.

2.5. Assessment

An assessment step is required to quantify the performance of the QbSH system. Mean Reciprocal Rank is the most extended metric and assesses almost every QbSH system currently

2.5.1. Mean Reciprocal Rank

Mean Reciprocal Rank (MRR) is a statistic for assessing any system that produces a list of possible responses to a query, ordered by similarity. The reciprocal rank of a query result is the multiplicative inverse of the rank of the first right response. So, if it appears in firs position the reciprocal rank is 1, and if it appears in third position, the reciprocal rank is 0.33 (1/3). The MRR is stated as the average of the reciprocal ranks responses for *n* queries.

$$MRR = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{rank(q_i)}$$

Therefore, a perfect QbSH 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.

3. Methodology / project development

This chapter describes all relevant methods that were utilized as well as research methods, measures and software development of the whole QbSH system. All the steps that the signal follows from the microphone of the phone, to the server and back as results to the phone are described in this chapter.

The followed procedure is based on the thesis of Pau Tur, but there are some differences in the implementation of the server, such the pitch extraction algorithm or some modifications in the pre-processing stage.

The whole implementation of the system as well the implementation of the client Android application is fully explained in the following sections.

3.1. Block diagram of the system

The following diagram shows in detail the main stages of the QbSH system developed in this thesis.

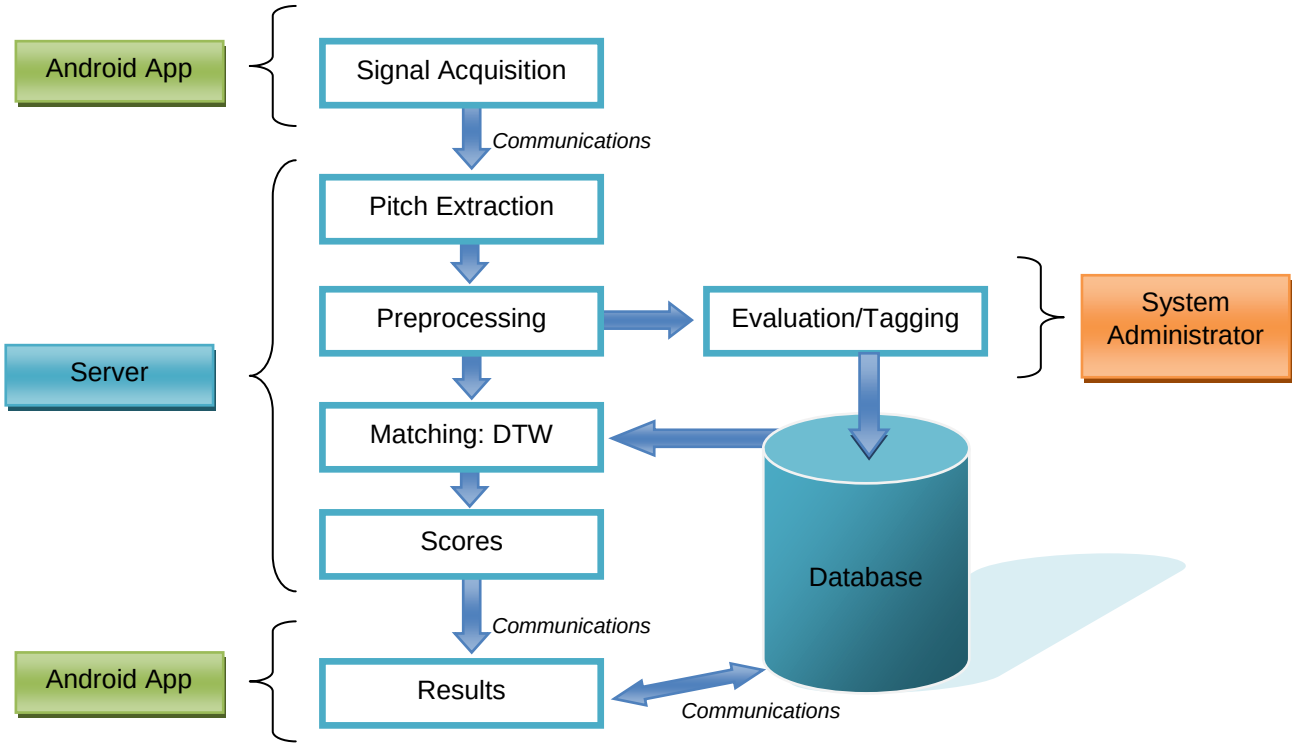


Figure 2: Block Diagram

3.2. Signal acquisition and transmission

This system of QbSH is based on the queries recorded by the user, and the database is created from these reviewed and tagged queries.

The acquisition process is done for all the queries using the Android app through the built-in microphone in the mobile phone. The audio is recorded with the AudioRecorder library from the Android SDK with a sample frequency of 8000 Hz and 16 bits for sample.

In the app screen (*Figure 18*) appears a microphone button and while it's pressed, the mobile is recording and fragmenting the audio in windows of 512 bytes and this fragments are sent in real time to the server through a socket connection. When the button is released the mobile stops recording and ends the connection to the server with an exit code.

3.3. Pitch extraction

In order to obtain the melody from the recorded query, the next step is to extract the pitch, that is, the sequence of frequency values that define that melody.

The pitch extraction algorithm used in this thesis is a python script that uses the tkSnack library and it's called as a system call from the C++ server code. It's called with a minimum frequency of 50Hz and maximum of 900Hz and with the RAPT (or AMDF) method (explained in chapter 3.3.1).

The main difference with the Pau Tur's project is that he used a polyphonic pitch extractor because he tried to extract the melody from the original songs. The purpose of this thesis is to compare queries with queries in order to obtain better matching score. So, the input is a monophonic audio file and the pitch extraction it's done with an algorithm optimized for monophonic human voice signals.

In the next figure (Figure 3) it's shown the output of the pitch extraction algorithm with the song "You never can tell" by Chuck Berry.

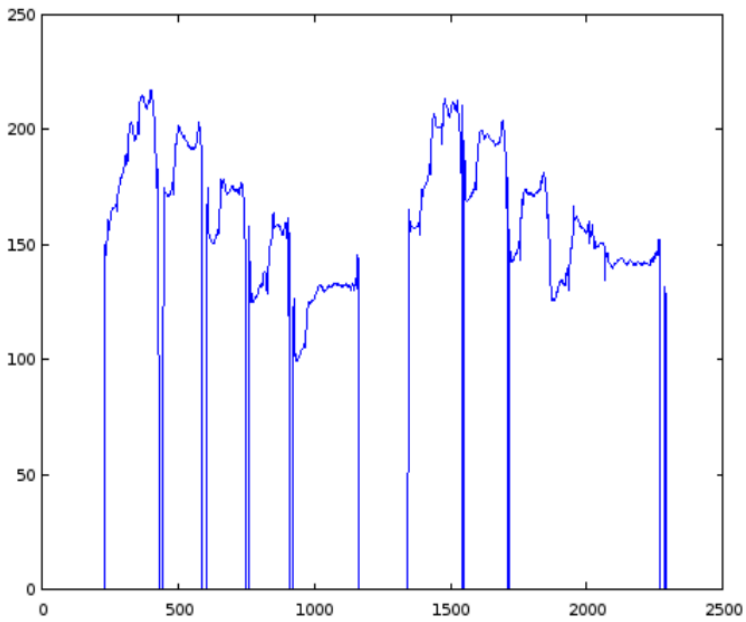


Figure 3: Pitch extraction example

3.3.1. Algorithms

The pitch detector/tracker presented here is a Robust Algorithm for Pitch Tracking (RAPT). The RAPT algorithm to be described is based on the normalized cross-correlation function (NCCF).

Given x_n a sampled speech signal with sampling interval $T = 1/F_s$, analysis frame interval *t*, and analysis window size *w*, at each frame we advance $z = t/T$ samples with $n=w/T$ samples in the autocorrelation window.

Then, the NCCF of K samples length, K < n, is defined as:

$$\phi_{i,k} = \frac{\sum_{j=m}^{m+n-1} x_j x_{j+k}}{\sqrt{e_m e_{m+k}}}$$

$$\text{where } k = 0, K-1;$$
$$\text{where } m = iz;$$
$$\text{where } i = 0, M-1;$$
$$\text{where } e_j = \sum_{i=j}^{j+n-1} x_i^2$$

Here is an overview of the steps that constitute RAPT:

- Provide two versions of the sampled speech data; one at original sample rate; another at a significantly reduced rate.
- Periodically compute the NCCF of the low sample rate signal for all lags in the F0 range of interest. Record the locations of local maxima in this first-pass NCCF.
- Compute the NCCF of the high sample-rate signal only in the vicinity of promising peaks found in the first pass. Search again for local maxima in this refined NCCF to obtain improved peak location and amplitude estimates.
- Each peak retained from the high-resolution NCCF generates a candidate F0 for that frame. At each frame the hypothesis that the frame is voiced/unvoiced is also advanced.
- Dynamic programming is applied to select the best set of NCCF peaks or voiced hypothesis at each frame based on a combination of local and contextual evidence.

Another pitch detector used in this thesis is a type of the Average Magnitude Difference Function (AMDF) detectors, or also called comb filter methods.

This detector is simpler and does not work as well as the RAPT method, but because it was tried in this thesis, a brief overview is given.

The AMDF pitch detector forms a function which is the compliment of autocorrelation function, in that it measures the difference between the waveform and a lagged version of itself.

The signal is windowed in order to obtain the pitch locally for each window, and to have evolution thorough the whole signal. The generalized function applied to each window frame is:

$$s(\tau) = \int_{-\infty}^{\infty} |x(t) - x(t+\tau)|^b \, dt, \qquad \text{where } b = 1$$

And fundamental frequency is the smallest period value taken as:

$$\widehat{f_0} = \frac{1}{\tau_{min}}, \quad \text{where } s(\tau_{min}) = \min_{\tau} s(\tau), \ \tau > 0$$

For discrete signals, our case, the AMDF function is given by:

$$s(\tau) = \frac{1}{N} \sum_{n=0}^{N-1} |x(n) - x(n+\tau)|^k, \quad \text{where } k = 1$$

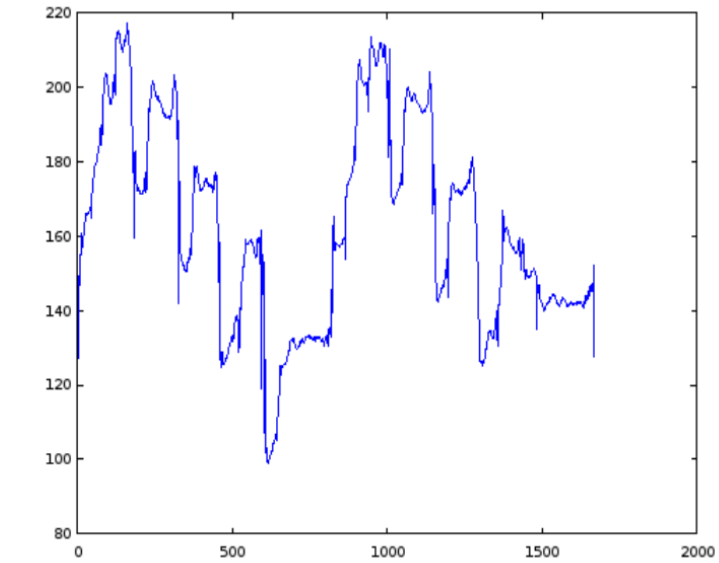
3.4. Pre-processing

The pre-processing 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.

The next step applied to the signal in some systems is the silence removal. In the extracted pitch sequences, the silences are represented with a frequency equal to zero, so the silence removal step is just to remove the frequencies equal to zero.

If we think about this step it does not seem logical to remove silences, because in order to recognize a song, the pauses among notes will certainly be as important as the notes themselves. However, if we compare results (Point 5), it turned out to be better when silences have been removed.

In the next figure (Figure 4), we can observe the pitch contour after applying silences removal to the signal plotted in figure 3.



.

Figure 4: Silences removed

After that, the signal looks very noisy, so the next step attempts to smooth the signal. To do so, a median filter of size 7 has been applied.

The median filter runs through the signal sample by sample, replacing each sample with the median of neighbouring entries. The most obvious window is just taking the first few preceding and following samples. The *median* of the finite list of samples included in the window can be found by arranging all the observations from lowest value to highest value and picking the middle one. If there is an even number of observations, then there is no single middle value; the median is then usually defined to be the mean of the two middle values.

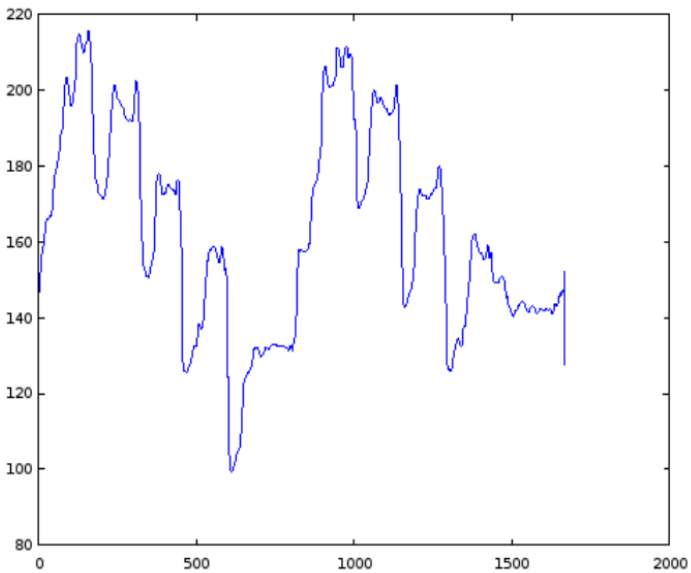


Figure 5: Median filter applied

In order to decrease the computational load of the system, making it faster to accomplish the specifications numbered in chapter 1.2, there is a need to reduce the number of samples of the signal. To keep the signal with a good resolution and to reduce the number of samples, a downsampling function is performed. This step returns a signal with one sample for every three samples in the original signal.

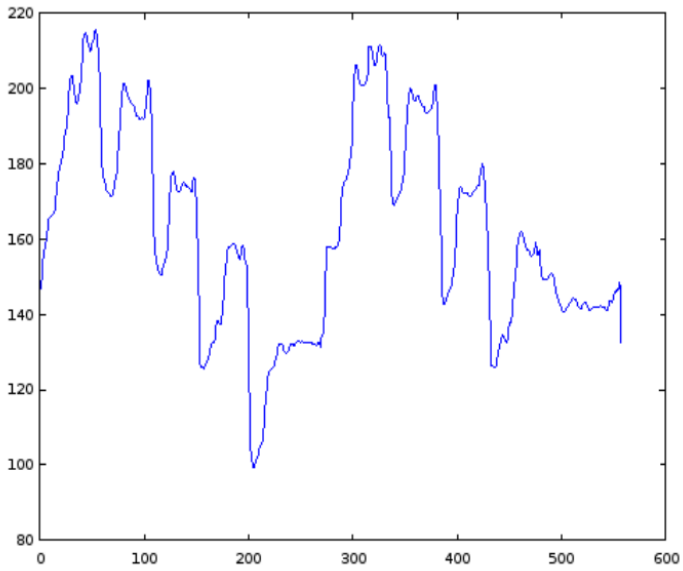


Figure 6: Downsampled signal

For computational analysis, melody is represented typically by the pitch but for a musically meaningful representation, we convert f0 from Hz to MIDI notes. This is a very important step because the step among notes is normalized, that is, a step of one in the MIDI scale is a semitone change. We are transforming from a non-linear scale (pitch, frequency in Hz) to a linear scale. The formula to convert frequency values into MIDI notes is:

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

The quantity $\log_2(\frac{f}{440})$ is the number of octaves above the tuning frequency of A4 = 440Hz (it is negative if the frequency is below that pitch). Multiplying it by 12 gives the number of semitones above that frequency. Adding 69 gives the number of semitones above the C five octaves below middle C (261.62 Hz).

Another possible map is to convert the pitch from Hz to Cent. Cent is also a logarithmic scale which is used to express a ratio between two frequencies, but it also can be used as a scale using 55.0 Hz as a reference frequency:

$$f_{0,cents} = 1200 * \log_2(\frac{f_{0,Hz}}{55.0})$$

This scale represents with more resolution the pitch changes because 100 cents correspond to 1 semitone (1 step in MIDI). The smallest resolution than an average human can hear is a 1/20 semitone, that is 5 cents.

Frequency (f_0) Hz	*MIDI*	*Frequency ($f_{0,\ cents}$)*
5	-9	-4151
8.176 (C1)	0	-3299
55 (A1)	33	0
440 (A4)	69	3600
880 (A5)	81	4800
1000 (1kHz)	83	5021

Table 1: Frequency conversions

The previous table shows the difference between the MIDI conversion and the Cents conversion. We can observe that if we double the frequency (we raise an octave), in the midi value we should add 12, the 12 semitones that are in the octave; and in the Cents value we should add 1200.

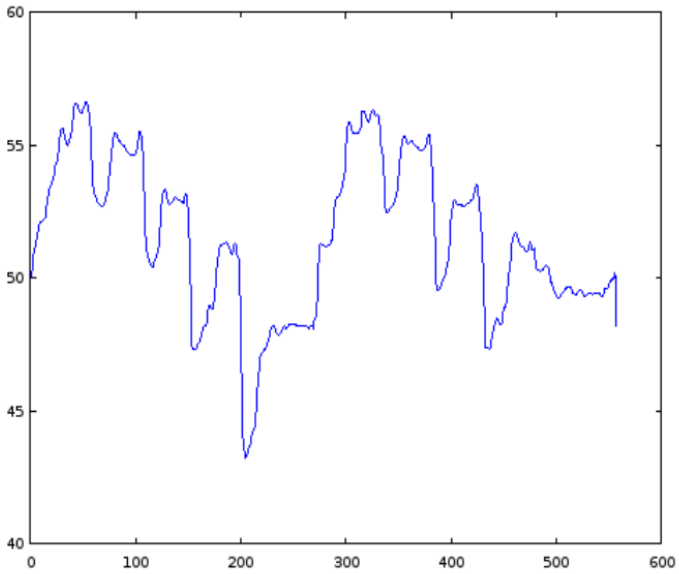


Figure 7: Midi representation of the melody

The next step is to segment the stable note regions in the midi sequence. That is because we are concerned with melody and the most important aspects of a melody are the notes. In this section we try to find those notes as stable regions in the signal. To do that we apply the following algorithm:

1. At each sample (for each midi value) the standard deviation of the past #winsize midi samples is computed.
2. A deviation threshold is applied to determine if the current frame belongs to a stable note region or not. Since, we are interested in the stable note regions, the standard deviation of the previous #winsize number of midi samples should be less than the deviation threshold.
3. All consecutive frames belonging to the stable note regions are grouped together into segments.
4. The segments which are smaller in duration than a threshold (minimum note duration) are removed.

There are two possible implementations of this algorithm. The first (method a) is to assign to each sample the mean of the frame centred on it. The second way (method b) is to assign to all consecutive frames belonging to a stable note region the same mean (the mean of all of them). We can see the difference in the following figures:

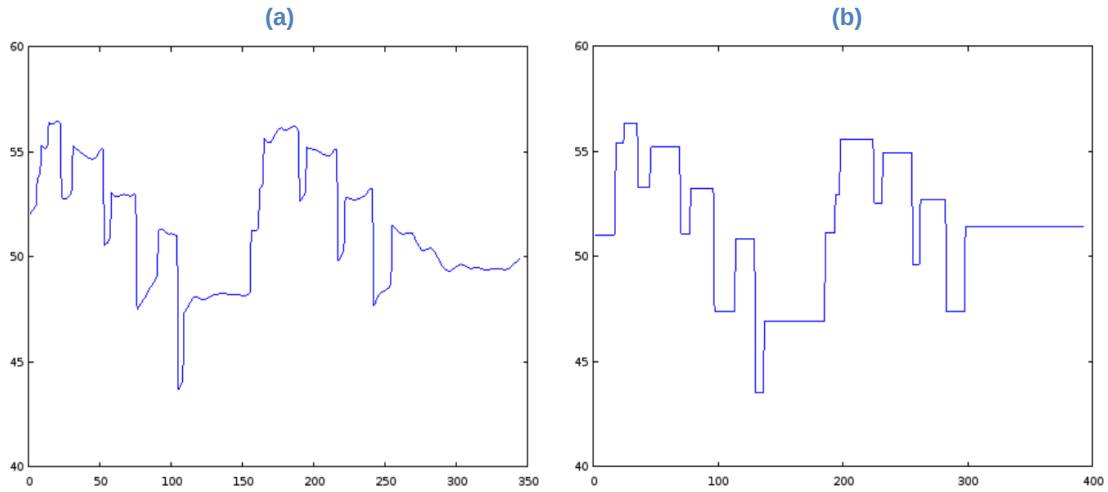


Figure 8: Stable note regions

In figure (a), each frame is assigned with its mean, and in figure (b) all consecutive stable note frames are assigned with the same mean. We can see that with this process a lot of noise is reduced, and this will be crucial for the next step.

In the second method the detailed information about the pitch is lost, but in order to compare songs there is no sense in comparing these little details, but the general pitch contour.

Finally, in order to remove tonality changes between different queries of the same song, the difference between the current sample and the previous one is computed. This allows us to remove the key in which the song has been sung and compare all melodies without this problem. After this step the signal is ready for the matching step.

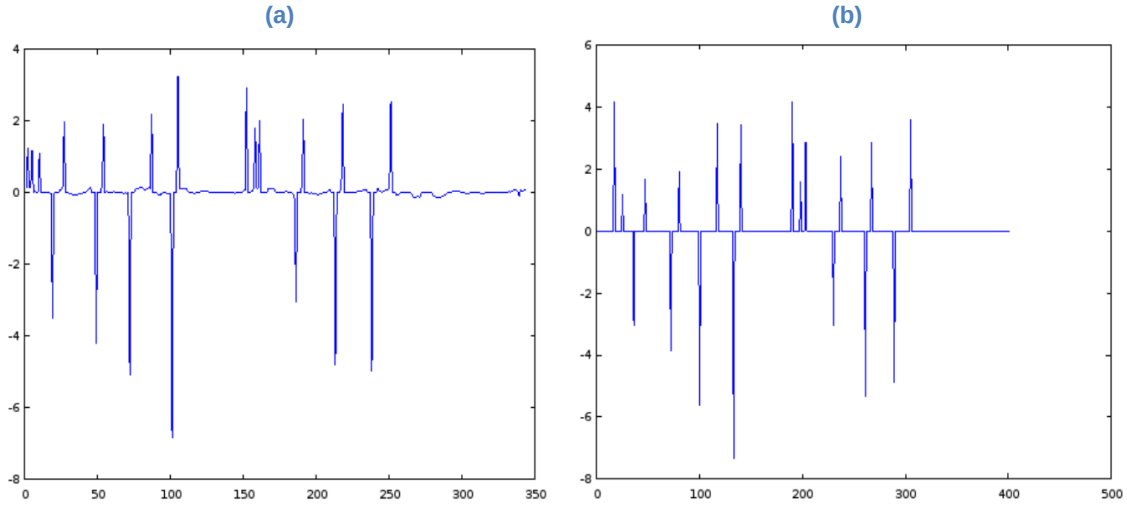


Figure 9: Differences signal

The plots above show the differences signal referred to both ways to implement the stable note regions detector. The horizontal axis represents time and the peaks in the vertical axis represent the number of semitones changes between notes. As we can see, we can considerably reduce the noise on this signal by assigning the same mean (method b) to each stable note region.

3.5. Matching: Dynamic Time Warping

Once the signal has been acquired, sent to the server, processed, and the differences signal has been obtained, the final step is the matching. That is, the comparison between the query signal and every signal in the database to find the best match.

In the processing stage the problem of different tonality queries was solved with the difference signal. In order to match two queries the time variations between signals must be taken into account.

The Dynamic Time Warping (DTW) algorithm is very useful for a Query by Humming system, since it allows the comparison between two signals removing the time variations between them.

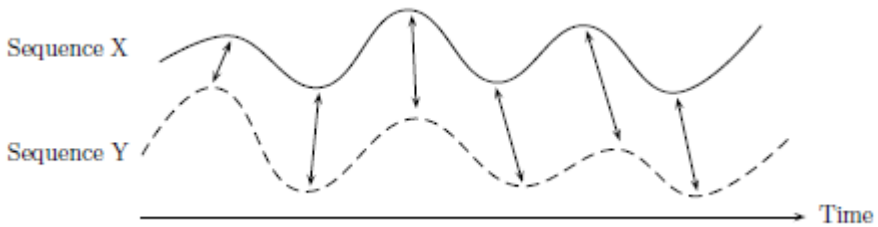


Figure 10: Dynamic Time Warping (Müller.M, 2007)

To compare two different signals x and y, a local cost measure (also local distance measure) is needed. This cost function c(x, y) is small (low cost) if x and y are similar, and otherwise is large (high cost).

Evaluating this cost function for each pair of elements of the sequences X and Y, the cost matrix C is obtained, defined by $C(n,m) = c(x_n, y_m)$. Then the goal is to find an alignment between X and Y having minimal overall cost. (Figure 11)

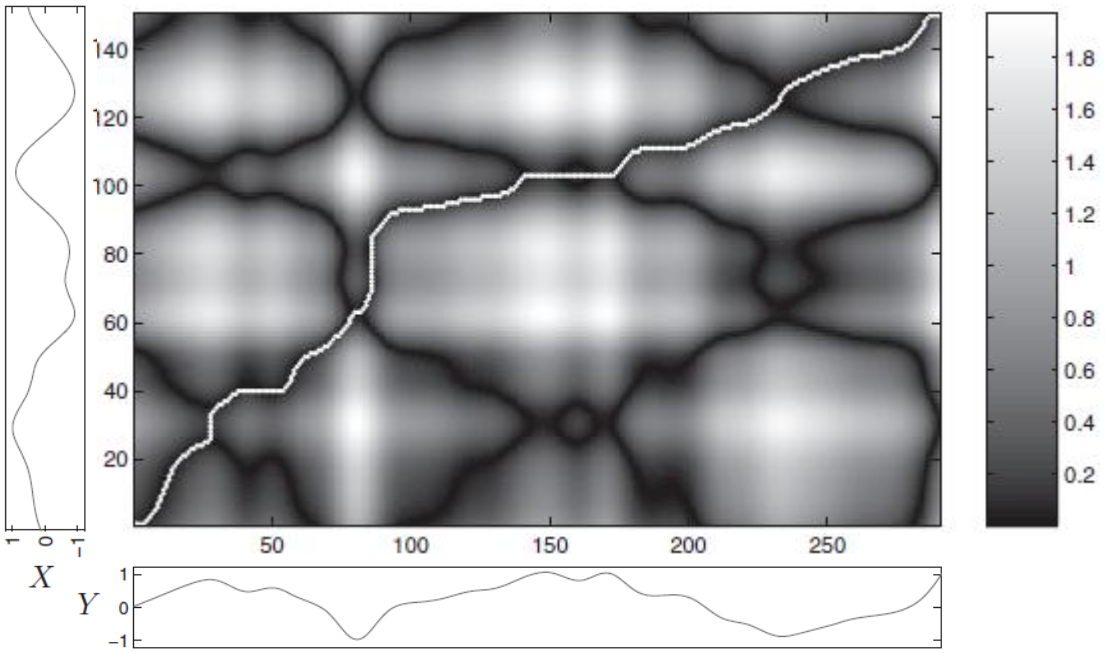


Figure 11: Cost matrix C with optimal warping path (Müller.M, 2007)

The total cost cp(X,Y) of a warping path p between X and Y with respect to the local cost measure c is defined as:

$$c_p(X,Y) = \sum_{l=1}^{L} c(x_{n_l}, y_{m_l})$$

Furthermore, an optimal warping path between X and Y is a warping path p* having minimal total cost among all possible warping paths. The DTW distance DTW(X,Y) between X and Y is then defined as the total cost of p:

$$DTW(X,Y) = c_{p*}(X,Y) = min\{c_p(X,Y) \mid p\ is\ an\ (N,M) - warping\ path\}$$

To determine an optimal path p, one could test every possible warping path between X and Y. Such a procedure, however, would lead to a very large computational complexity. To avoid that, an algorithm based on dynamic programming is applied. By definition:

$$D(i,j) = DTW\big(X(1:i), Y(1:j)\big)$$
$$\text{where } X(1,i) = (x_1, \dots x_i)$$
$$\text{where } Y(1,j) = (y_1, \dots y_j)$$

The values D(i,j) define an NxM matrix D, which is also referred to as the accumulated cost matrix. Then, the optimal warping path p* is easier to compute:

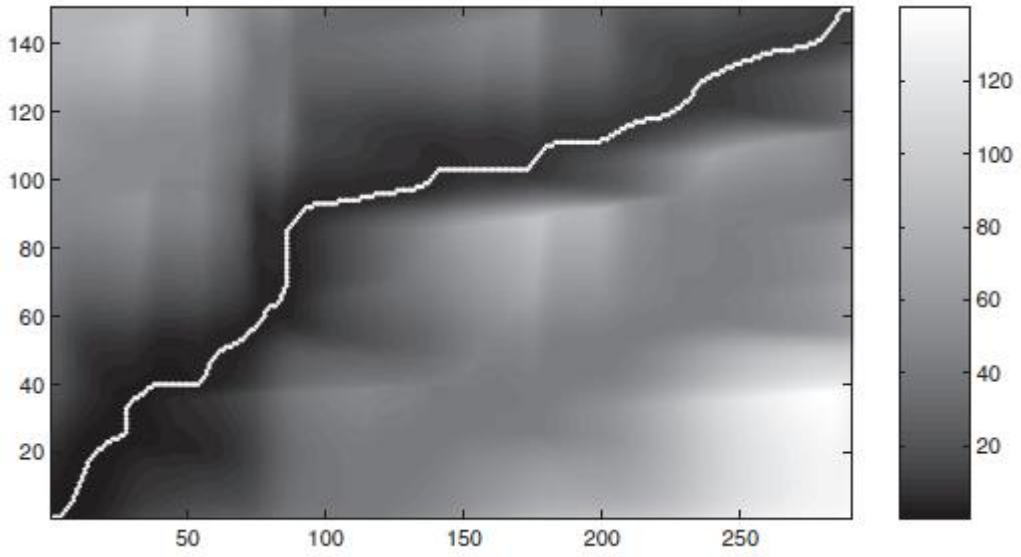


Figure 12: Accumulated cost matrix D with optimal warping path (Müller.M, 2007)

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. However, this option has an important drawback: it allows horizontal and vertical paths.

As can be seen in figure 12, 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 second DTW path (figure 13).

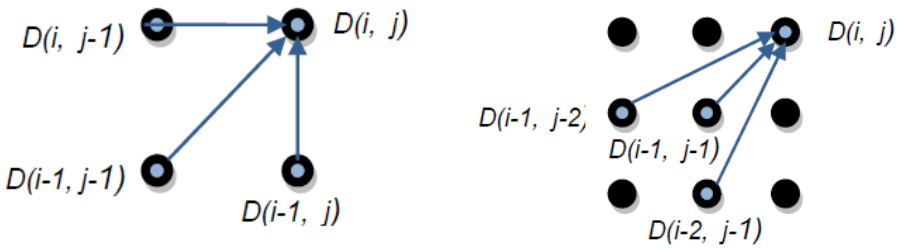


Figure 13: First and second DTW paths configuration

In this way, having only diagonal paths avoids mapping a single sample with many others. As can be seen in the second DTW path configuration, the path is able to grow at double, half or exactly the same speed amongst the length of the signals. But this path configuration has another problem: it can skip samples in columns or rows, that is to say, some of the samples of the signals are ignored if the related cost it has been too high.

Finally, an alternative option has been used (Figure 14). It allows vertical and horizontal moves within a single sample step, but forces the full path to grow diagonally. 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.

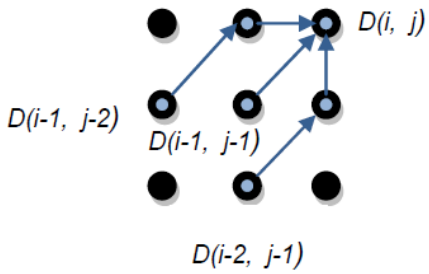


Figure 14: Final DTW path configuration

Another point to bear in mind is the fact that the original DTW algorithm tries to match the two signals from start to end. In the QbH systems, this makes no sense, since we are trying to match a 10-second-length query with other signals that can be shorter or larger than the query itself. For example, if a query contains the chorus of a song, the algorithm should allow the matching process to start from any point in the song, avoiding the initial part of it.

To fix that, the algorithm has been modified to allow the start and end samples of the query to be matched from any start point in the song.

As it can be seen in the figure below, the path can start and end anywhere, and it always grows diagonally in order to avoid sample skipping and horizontal or vertical paths.

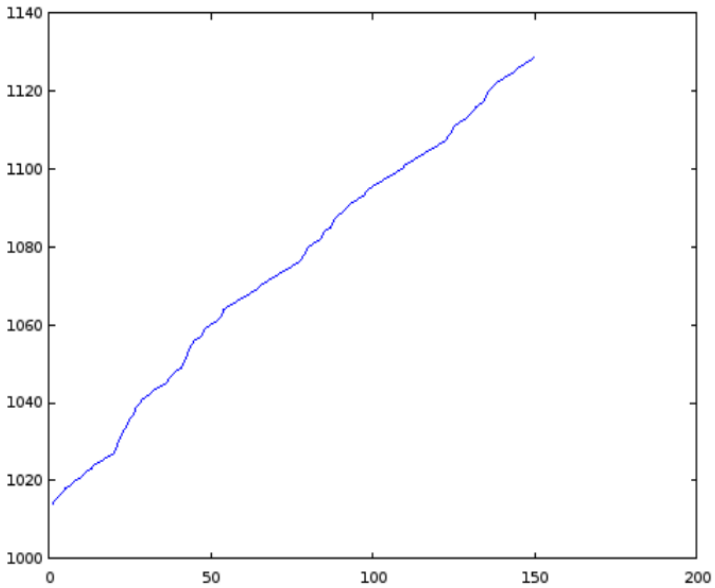


Figure 15: Path from the matching of a two signals

From the viewpoint of computational load, this stage demands the greatest effort. A basic DTW implementation written in C code has been used as base code, and modifications have been made to it in order to fix all the problems commented above.

4. The application

In this section, a review of the Android application developed is discussed. The Android application (from now on *the* app) has been developed as a client for the server and as a prototype opened for future developing work. The general procedure is very simple: there are three possible methods to use the app:

- *The standard mode*: The normal way to use the app, a query is recorded and the results of the matching process are shown in the screen.
- *The guided mode*: In order to contribute to the database growth, the user is asked to sing a concrete song.
- *The non-guided mode*. In order to contribute to the database growth, the user is asked to sing any song and then is asked to the name and artist of the song.

It has been written in Java using the latest Android SDK (SDK 21), but is compatible with all SDK versions higher than SDK 9. In order to work properly, the app needs user permission to access Internet, to access the network state and to access directly to the microphone of the phone.

4.1. User registration/Login

A user registration/login service has been developed in order to control the use of the application, especially for the guided and no-guided modes. This is to control who is recording queries to increase the database, to sort and filter the database for users and also to develop other uses in the future.

It is a very simple registration/login screen only with the name and password. This pop-up dialog only appears the first time the app is opened (or in the case that the data of the app had been erased by the user).

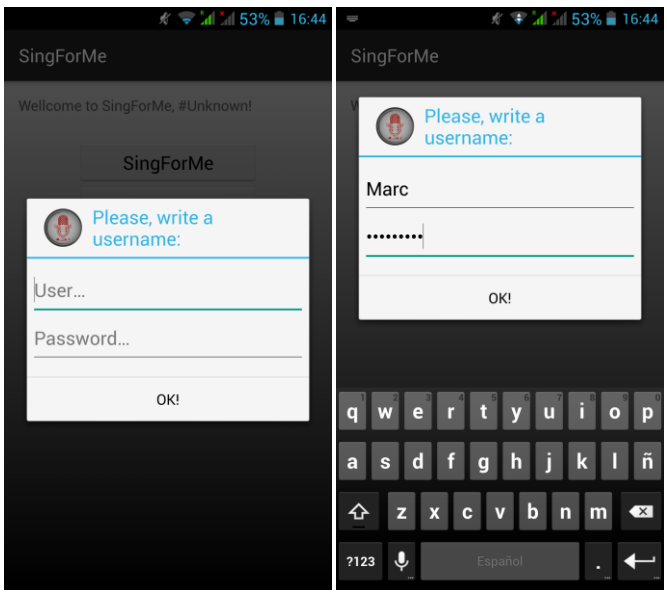


Figure 16: User registration pop-up

The data entered in the boxes is sent to the server and it searches in the user's database if the user credentials received exists. There are three possible answers from the server:

- The user exists and the sent password is correct. The application proceeds to log in the user and the main screen appears.
- The user exists and the password is incorrect. The user is asked to reinsert the password and try to login again.
- The user doesn't exist The application proceeds to login the user, the main screen appears, and the server registers the user's data into the user's data base.

4.2. Main Screen

The main screen offers the user the choice of what he wants to do with the application. As can be seen in the next figure, there are four buttons. One for each mode the application can work and one more that shows information about the use of the app.

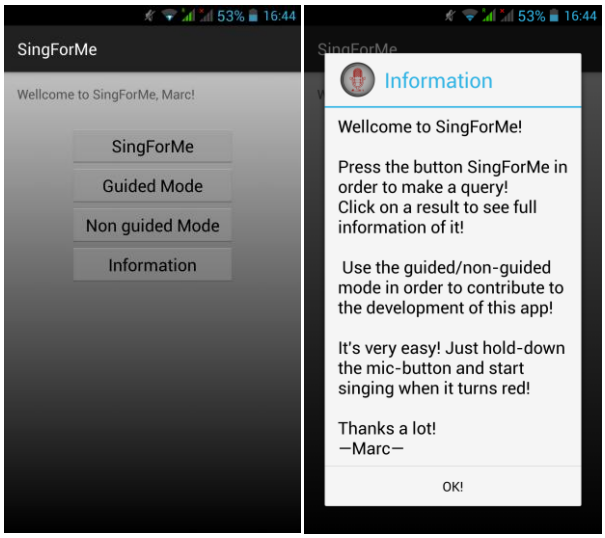


Figure 17: Application main screen & Information popup

4.3. Standard Mode

The standard mode procedure is very simple and follows the system as described above. The app records a query through the mobile microphone and once the results have been computed and sent back from the server, the top-10 list is shown.

When the standard mode is selected in the main screen of the app, a very simple screen shows up with a mic-button. In the figure below the mic-button is shown on its three possible states: while it is not pressed (a), while it is pressed (b & c)

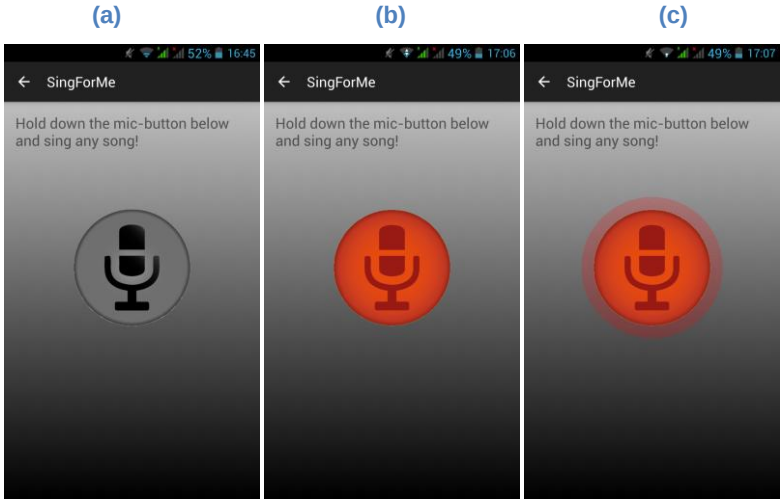


Figure 18: Mic button

When the mic-button is pressed, the app sends to the server a first package with the user's info and the mode in which is working. When the app receives the OK package from the server, the mic-button turns orange (and starts a blink animation) and starts recording from the mobile's microphone.

As explained in chapter 3.1, the audio is recorded with the AudioRecorder library from the Android SDK with a sample frequency of 8000 Hz and 16 bits for sample. The audio is being fragmented in real time in windows of 512 bytes and these fragments are being sent continuously to the server through a socket connection. When the button is released the mobile stops recording and ends the recording connection to the server with an exit code.

While the app is waiting for the results from the server, a progress dialog with a spinning wheel appears. When the results are correctly received, this dialog disappears and the results screen is shown.

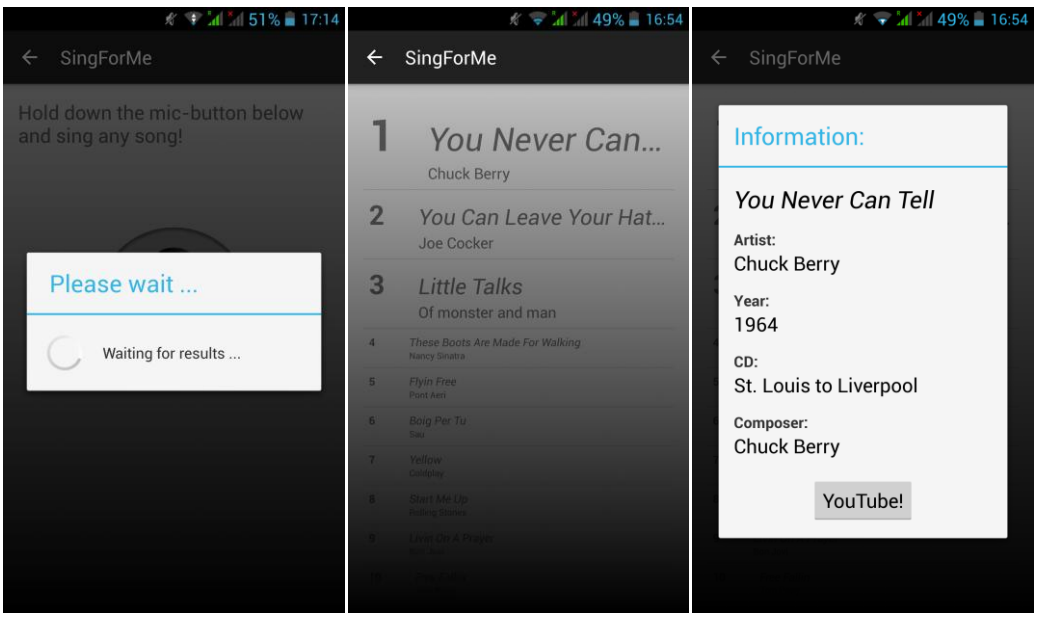


Figure 19: Results in the app

By clicking on any item on the list of results, a popup with information about the song will appear (Figure 19). Also there is a button that redirects to the clicked song on YouTube[2].

Every click in a song is registered in the server an could be used in future development in order to auto-tag and auto-include the queries in the database.

4.4. Guided / No-guided modes

These two methods have been developed with the single purpose of creating and adding queries to the database. In the guided mode, a name of a song is given to the user in order to record a query for this song and store it on the database. The non-given mode is similar to the previous one, but the user sings any song and is asked to give the name of the song, in order to be stored in the database.

In the guided mode, before the mic-button appears (figure 13), a pop-up shows up with a random song that the user is asked to sing. The user can accept it or select randomly another song. Once the song is selected and recorded it is sent (with the song info) to the server and stored in a different folder than the ones from the standard mode.

In the no-guided mode, the user is asked to record a song (with the same mic-button screen Figure 13) and when he has finished, a pop up shows up to ask him which song he has sung.

Once the song info is entered into the boxes, it is sent to the server and stored with the previously recorded song.

Guided Non-Guided

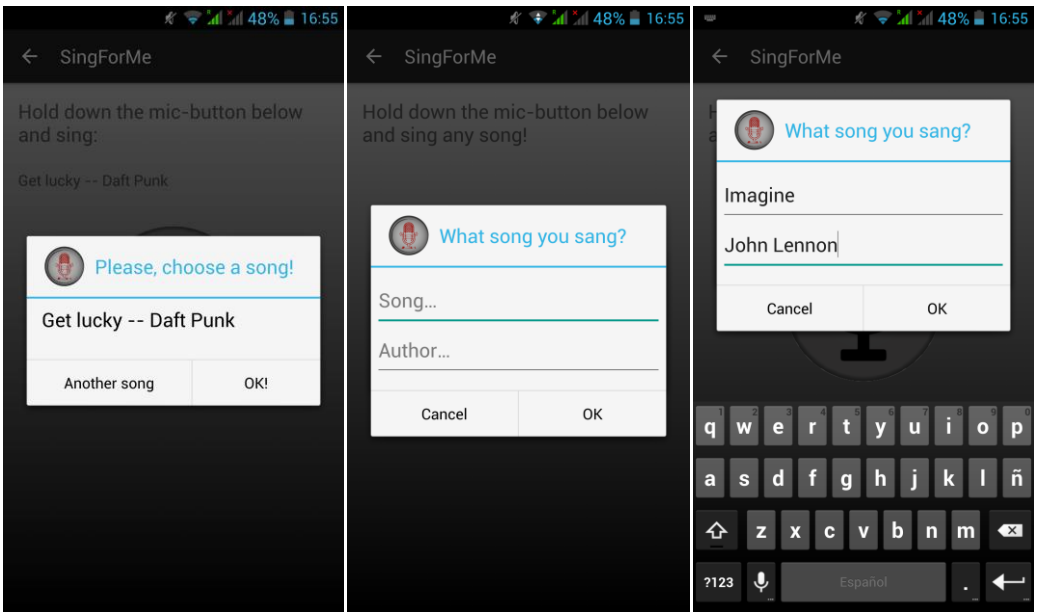


Figure 20: Guided / No-guided mode

[2] www.youtube.com

5. Results

Once the system is built, and all its parts are correctly working, several offline and online tests have been executed to assess the performance of the complete program.

These tests try to measure the performance of the server, the performance of the application and of course the performance and results of the whole QbH system.

To do that, the database has been divided in train and test subsets. As explained before, a DTW algorithm to find the best match does not need to be trained, so the meaning of those two datasets is more like target (train) and queries (test).

For the evaluation of the complete test database, the 10 results with minimum distance (cost) in the matching process will be given, and the Mean Reciprocal Rank (MRR) will be computed (*view 2.5.1*.).

5.1. The database

The database used has been created especially for this purpose and for this project. This database is formed by queries recorded during part of the duration of the system implementation.

This database[3] contains 300 queries (on 31/01/2015), which are divided in train (target) database (250 queries) and a test (query) database (50 queries).

5.2. Parameters

In order to assess the system, the database has been randomly divided and all the following parameters have been tuned.

Many experiments have been done in order to tune these parameters and once the best combinations have been found, the database has been randomly divided (different division than the first one) in order to compare results and fine tune a little more. The best set of parameters used is listed in the table below:

Removing Silences		Yes
Median filter size		3
Downsampling factor		3
Midi scale/F_{cent}		Midi
Segmentation Of Stable Note Regions	Window Size	5
	Thld	1
	Std. Thld	0.3
Median filter size		5

Table 2: Best set of parameters

[3] Fully detailed in Appendix 1

5.3. Best results

The following table represents some of the most important results, showing how the MRR changes with some big changes in the parameters.

Recall that a perfect system that always retrieve the correct song in the first time has a MRR = 1 and the worst system that never retrieves the correct song in a Top-X list has a MRR = 0.

All the tests containing changes in every particular parameter can be checked in the appendix section at the end of the document.

As stated previously, the data base has been divided twice in order to compare results and have a more accurate MRR score.

Parameters	MRR(first split)	MRR(second split)
RAPT + Silence Removal + mf (3) + downsampling (3) + Seg.StableRegion (5,1,0.3) + mf (5)	0.743100	0.757619
AMDF + Silence Removal + mf (3) + downsampling (3) + Seg.StableRegion (5,1,0.3) + mf (5)	0.428150	0.529894
RAPT + mf (3) + downsampling (3) + Seg.StableRegion (5,1,0.3) + mf (5)	0.697889	0.739683
RAPT + Silence Removal + mf (3) + downsampling (3) + Seg.StableRegion (5,1,0.5) + mf (5)	0.658667	0.73333
RAPT + Silence Removal + downsampling (3) + Seg.StableRegion (5,1,0.3) + mf (5)	0.681524	0.736825

Table 3: Representative results of the assessment of the system

We can observe, as said in chapter *3.4*, that if we keep the silences, the results are less accurate than if we remove them and that the RAPT method outperforms the AMDF method for pitch extraction. Also, if we eliminate a step or we change a little the values of the parameters this produces noticeable changes in the MRR result.

We can also see that there is not a big difference between both different splits of the database, so we can confirm that the result is robust and reasonable.

6. Budget

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

All the software used in this project (Eclipse, Android SDK, pitch extraction script and DTW libraries) is free source and is described throughout the document.

	Quantity	Price	Cost
Supervisor (wage)	46 hours	25€/hour	1150€
Junior Engineer (wage)	690 hours	8€/hour	5520€
Computer for developing (amortization)	1	5.2€/week	120€
Computer as server (amortization)	1	5.2€/week	120€
		TOTAL	6910€

Table 4: Estimation of the total cost of the project

7. Conclusions and future development

In this project a whole QbH system has been developed in a very realistic way as a client/server system. Some state-of-the-art techniques have been analyzed and applied to accomplish the development of the system.

In order to improve the results and because of the multi-user interaction with the system, query files have been used as reference signals. The rationale is that those signals were very easy to acquire once the application was fully working, because the queries made in the application are stored in the server in order to be added in the database.

I think that there is a lot of work to do in this section, in the automatic insertion of the songs in the database. That is, for the songs to be automatically and correctly added in the database, rejecting those bad queries (with a lot of noise, not well sung, etc) and adding automatically the metadata to each non-previously stored query.

A very good pitch extraction algorithm for monophonic signals has been used in the feature extraction step, once in the server. After this pitch extraction many processing techniques have been tried and analyzed to prepare the signal for the matching stage. Based on the experimental results, not removing the silences, the size of the median filters, and the note segmentation step improved the performance of the system in a very noticeable way.

The Dynamic Time Warping algorithm has been used to compute the distance between the queries and the database giving very good results in the matching step.

Once the server was designed and working, a complete Android app has been developed, making the system accessible from any Android device. The app allows the user to record a fragment of voice and search for the best match in the database. With the application, the user can also contribute to increasing the size of the database, by recording queries.

The transmission of the audio has been done by streaming in real time, and each connection to the server starts a new thread, so the server can accept various users connected at same time.

The execution time of the whole process is very low, around 10 seconds since the song has been recorded till the results are shown on the mobile screen. Although this is a very low execution time, there is clearly a lot of room for improvement in every stage of the whole system in order to further reduce the execution time. A possible and easy way to improve the system could be computing all the DTWs in parallel, since the DTW computation demands the greatest effort of the whole system.

The obtained results are very good, achieving a MRR around 0.75 in the 300 varied songs database.

Reviewing the system requirements and specifications we can see that all the requirements have been achieved. The only point not achieved is the size of the database, and this fact may change slightly the results in the time of execution and in the general MRR.

Query by Singing/Humming as said before, is still a relatively recent topic with many more research groups than commercial systems.

Bibliography

[1] P.Tur, “Query by Humming”, Degree's Thesis, Escola Tècnica Superior d' Enginyeria de Telecomunicacions de Barcelona, June 2014

[2] T.C.Nagavi, B.U. Bhajantri. "An extensive analysis of query by singing/humming system through query proportion", The International of Multimedia & Its Applications (IJMA), Vol 4, No.6, December 2012

[3] Müller, M. "Information Retrieval for Music and Motion", Springer Berlin Heidelberg, Part 1 pages 69-84, ISBN 978-3-540-74047-6, 2007

[4] M. J. Ross, H. L. Shaffer, A. Cohen, R. Freudberg and H. J. Manley, "Average Magnitude Difference Function Pitch Extractor," IEEE Trans. on Acoustics, Speech and Signal Processing, vol. 22, no. 5, pp. 353-362, 1974.

[5] Antonelli, M., Rizzi, A., & del Vescovo, G. "A Query by Humming System for Music Information Retrieval". 10th International Conference on Intelligent Systems Design and Applications ISDA'10. Cairo, Egypt 2010

[6] P.Papiotis "Real-time accompaniment using lyrics-matching in a Query-by-Humming system", Master on Music Computing thesis, Universitat Pompeu Fabra, 2010.

[7] Guo, Z., Wang, Q., Liu, G., & Guo, J. "A music retrieval system using melody and lyric", IEEE International Conference on multimedia and Expo Workshops ICMEW. Melbourne, Australia 2012.

[8] 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, 2013

[9] Guo, Z., Wang, Q., Yin, L., Liu, G., & Guo, J. (2012). "Query by Humming via Hierarchical Filters". 21st International Conference on Pattern Recognition ICPR. Tsukuba, Japan 2012.

[10] Yi Ma, "Query by Humming using HMM-based Approach", Term project on CSE 788.J04, The Ohio State University. Spring 2012

[11] Chen, A., Chang, M., & Chen, J., "Query by Music Segments: An Efficient Approach for Song Retrieval". Proceedings of IEEE International Conference on Multimedia and Expo ICME. New York, USA 2000.

[12] R. Meyer, "Professional Android 4 Application Development", Wrox, 3ºEdition, May 2012.

[13] Z.Mednieks, "Programming Android: Java programming for the new Generation of mobile devices", O'Reilly Media, Second Edition, October 2012.

[14] *Midomi - SoundHound*. November 2015 from http://www.midomi.com/

[15] *Musipedia* November 2015 from http://www.musipedia.org/

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

[17] Talkin, D. "A Robust Algorithm for Pitch Tracking (RAPT)". In Ch. 14 in Speech Coding and Synthesis, 1995.

Appendix I: Database

This appendix contains a list of all the songs[4] that the database contains. For each song there is a metadata file (.txt) and a few recorded queries (.wav), from the same or different user. For each query there are 4 files: the audio file (.f0), the pitch file (.f0), the processed pitch file (.proc) and the differences file (.diff).

Song title	Author	Number of queries
Ain't no mountain high enough	Marvin Gaye	2
Al Mar	Manel	2
All my loving	The Beatles	9
All Star	Counting Crows	3
All you need is love	The Beatles	3
Another brick in the wall	Pink Floyd	2
Another day in paradise	Phil Collins	3
Another one bites the dust	Queen	3
Basket Case	Green Day	3
Boig per tu	Sau	4
Born in the USA	Bruce Springsteen	3
Born to run	Bruce Springsteen	3
Brown eyed Girl	Van Morrison	5
Californication	Red Hot Chili Peppers	2
Call me maybe	Carly Rae Jepsen	4
Can't buy me love	The Beatles	2
Carolina	M-Clan	3
Chandelier	Sia	2
Cum on feel the noize	Quiet Riot	3

[4] On 31/01/2015

Do you want to know a secret	The Beatles	2
Don't look back in anger	Oasis	2
Drive my car	The Beatles	2
Dust in the wind	Kansas	3
Everybody	Backstreet Boys	2
Feeling Good	Nina Simone	3
Final Countdown	Europe	4
Flyin' Free	Pont Aeri	3
Free Fallin'	Tom Peety	2
From me to you	The Beatles	2
Get Lucky	Daft Punk	3
Going Home	Dire Straits	2
Hallelujah	Leonard Cohen	3
Happy	Pharrell Williams	3
Happy Together	The Turtles	3
Hard to handle	Black Crowes	3
Have you ever seen the rain	Creedence Clearwater Revival	2
Here I go again	Whitesnake	2
Hey Jude	The Beatles	4
Highway to hell	AC/DC	4
Ho Hey	The lumineers	2
Home	Edward Sharpe	3
Hot Stuff	Donna Summer	3
House of the rising sun	The Animals	2
Hungry heart	Bruce Springsteen	3

Hush	Deep Purple	4
I am walking on sunshine	Katrina & the waves	3
I shot the sheriff	Bob Marley	2
I still haven't found what I'm looking for	U2	2
I wanna rock&roll all nite	Kiss	3
I want to break free	Queen	2
I want to hold your hand	The Beatles	3
I was made for loving you	Kiss	3
I will wait	Mumford & sons	4
Is this love	Bob Marley	2
Island in the sun	Weezer	3
It's my life	Bon Jovi	2
Jean-Luc	Els amics de les arts	2
La barbacoa	Georgie Dann	2
La flaca	Jarabe de palo	2
Let it be	The Beatles	3
Let it go	Frozen	4
Little tals	Of monster and man	3
Livin' on a prayer	Bon Jovi	3
Love me do	The Beatles	3
Money	Pink Floyd	3
Money for nothing	Dire Straigs	3
Obla-di, Obla-da	The Beatles	2
Para no verte más	La mosca	2
Por la boca vive el pez	Fito & fitipaldis	4

Redemption song	Bob Marley	3
Rolling in the deep	Adele	3
Satisfaction	The rolling stones	4
Seven nation army	The white stripes	3
Sex is on fire	Kings of Leon	2
She loves you	The Beatles	2
Smoke on the water	Deep Purple	6
Soldadito Marinero	Fito & Fitipaldis	2
Somebody that I used to know	Gotye	6
Son of a man	Phil Collins	3
Stairway to heaven	Led Zeppelin	6
Stand by	Extremoduro	2
Start me up	The rolling stones	3
Still got the blues	Gary Moore	3
Summercat	Billie The vision	6
Tears in heaven	Eric Clapton	3
The Fox	Ylvis	2
The sound of silence	Simon & Garfunkel	2
The trooper	Iron Maiden	3
These boots are made for walking	Nancy Sinatra	5
Three little birds	Bob Marley	2
Twist & Shout	The Beatles	2
Up where we belong	Joe cocker	2
Use somebody	Kings of Leon	2
Viva la vida	Coldplay	2

Walk of life	Dire Straits	7
We are the champions	Queen	3
What a wonderful world	Louis Armstrong	2
Whatever you want	Status Quo	3
Wicked Game	Chris Isaak	2
Wish you were here	Pink Floyd	3
Wrecking ball	Miley Cyrus	2
Yellow	Coldplay	4
Yellow submarine	The Beatles	4
Yesterday	The Beatles	6
You can leave your hat on	Joe Cocker	4
You never can tell	Chuck Berry	9
You're the one that i want	Grease	3

Table 5: Database details

Appendix II: Full Results

This appendix includes a table with all the tests that have been done and it's corresponding parameters. In the best cases with the first split of the database, a second split of it was made in order to check the results.

				segmentation stable					
Method pitch	silences removed	median filter	downsampling	window	thld	stdthld	median filter	**MRR split 1**	**MRR split2**
esps	yes	3	3	5	3	0.4	7	**0.653524**	
esps	yes	5	3	5	3	0.4	7	**0.665333**	
esps	yes	7	3	5	3	0.4	7	**0.646189**	
esps	yes	no	3	5	3	0.4	7	**0.635833**	
esps	no	no	3	5	3	0.4	7	**0.676667**	
esps	no	3	3	5	3	0.4	7	**0.684889**	
esps	no	5	3	5	3	0.4	7	**0.666714**	
esps	no	7	3	5	3	0.4	7	**0.660222**	
esps	no	3	5	5	3	0.4	7	**0.562191**	
esps	no	3	7	5	3	0.4	7	**0.539500**	
esps	no	3	3	5	3	0.4	7	**0.573410**	
esps	no	3	3	5	3	0.4	3	**0.697167**	
esps	no	3	3	5	3	0.4	5	**0.698024**	**0.715079**
esps	no	3	3	5	3	0.4	n	**0.689191**	
esps	yes	3	3	5	3	0.4	5	**0.649437**	
esps	no	3	3	3	3	0.4	5	**0.621889**	
esps	no	3	3	7	3	0.4	5	**0.673857**	
esps	no	3	3	5	1	0.4	5	**0.703056**	
esps	no	3	3	5	1	0.1	5	**0.324913**	
esps	no	3	3	5	1	0.7	5	**0.429357**	
esps	no	3	3	5	1	0.5	5	**0.723667**	**0.716327**
esps	no	3	3	5	2	0.4	5	**0.710079**	**0.733016**
esps	no	3	3	5	1	0.3	5	**0.697889**	**0.739683**
esps	no	3	3	5	2	0.5	5	**0.718222**	**0.711111**
esps	yes	3	3	5	1	0.3	5	**0.743100**	**0.757619**
esps	yes	3	3	5	1	0.5	5	**0.658667**	**0.733333**
esps	yes	no	3	5	1	0.3	5	**0.681524**	**0.736825**
amdf	yes	3	3	5	1	0.3	5	**0.428150**	**0.529894**

Table 6 : Full results