2019

Music Retrieval System Using Dynamic Time Warping

Emeka Jude Okafor

Music Retrieval System Using Dynamic Time Warping

Emeka Jude Okafor

North Carolina A&T State University

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Department: Computer Science

Major: Computer Science

Major Professor: Dr. Jingsheng Xu

Greensboro, North Carolina

2019

The Graduate College
North Carolina Agricultural and Technical State University

This is to certify that the Master's Thesis of

Emeka Jude Okafor

has met the thesis requirements of

North Carolina Agricultural and Technical State University

Greensboro, North Carolina
2019

Approved by:

Dr. Jingsheng Xu Major Professor	Dr. Xiaohong Yuan Committee Member
Dr. Jung Hee Kim Committee Member	Dr. Xiaohong Yuan Department Chair

Dr. Clay S. Gloster, Jr.
Interim Dean, The Graduate College

Biographical Sketch

Emeka Jude Okafor was given birth in Lagos, Nigeria. He earned his bachelor's degree in Computer Information Systems from Babcock University, Nigeria, in 2013. Emeka was known as a student leader and his excellent academic performance.

Emeka worked as a system analyst at Custodian Life Insurance, where he managed the implementation of the Business Enterprise Resource Planning System. He proceeded to obtain his master's degree in Computer Science at North Carolina Agricultural and Technical State University in the United States in 2018.

Dedication

Every challenging work needs self-efforts as well as guidance and love, especially from those close to our hearts. My humble effort, I dedicate to Almighty God for his grace and love. To my family, most notably my brother for being my guardian during my education.

Acknowledgments

I am very grateful to God for his guidance, protection, grace, and mercies through this journey. I thank North Carolina A & T State University, most notably the Department of Computer Science. I wish to acknowledge the effort of my advisor Dr. Jingsheng Xu for his support and suggestions during the process of this research. I want to express my gratitude to my thesis committee members: Dr. Xiaohong Yuan, who is also the graduate chair for her thoughtful advice and support, and Dr. Jung Hee Kim for her time and contribution to this research.

My sincere acknowledgment goes to my father, my mother, and my brothers, for their support, prayers, and encouragement.

My appreciation goes to Janet Osawere, whose love and prayers made me able to get such a success and honor. I am also grateful to Emmanuel Olalere for being a good friend and advisor. I am thankful to all my friends, who show sincere concerns towards my academic success. God bless you all.

Table of Contents

List of Figures

List of Tables

Abstract

With the growth of digital audio data, various and fast access to music data is strongly desired, especially for large music databases. A more natural way to retrieve a song from a database will be to hum to the tune. To relate and compare musical pieces is a very complex task. Musical compositions usually collapse multiple information sources and complex, multifaceted interactions established between parts. Despite such degrees of complexity, humans are outstandingly good at performing individual musical judgments with little conscious effort, while a computer cannot efficiently achieve this task.

In this work, we focus on one such task: music information retrieval using content-based input such as users' hum to tune a music piece. Query by humming (QBH) is another content-based retrieval method for Music Information Retrieval in an extensive music database. This method is a form of audio to audio mapping of events in one recording to the similar events in the other recording. In particular, we adopt computational approach information provided by the audio signal. We propose a system for music retrieval that matches a user humming to the best song in an audio database. We developed an algorithm to map the difference in two musical pieces in time series domain based on dynamic programming and longest subsequence. We explored alternative tunes for user query input, which might be off-tune, thereby achieving high accuracy on query identification from a music database. However, the accuracy of the result is highly dependent on the query quality and closeness to the actual song.

CHAPTER 1

Introduction

In this chapter, We introduced Music Information Retrieval and discussed the need for a new way to retrieve songs from a music database by using hum melody from a user as input. In this system, the user will hum the desired tune to a microphone and the system will use the user's hum melody as input to retrieve the desired target song.

1.1 Overview of Music Information Retrieval

To relate and compare musical pieces is a very complex task. Musical pieces usually collapse multiple information sources and several complex multifaceted interactions established between parts. In spite of such degrees of complexity, humans are outstandingly good at performing individual musical judgments with little conscious effort (Dowling & Harwood, 1985). Think for instance in the song "Happy birthday to you." If somebody sings its melody, even if some parts are out of tune, most listeners can easily recognize this musical piece provided that they are familiar with the song. Our research framed in the context of music information retrieval explores to achieve this result efficiently.

With the growth of digital audio data, various and fast access to music data is strongly desired, especially for large music databases. A more natural way to retrieve a song from a database will be to hum to the tune. Query by humming (QBH) is another content-based retrieval method for Music Information Retrieval in an extensive music database. This method is a form of audio to audio mapping of events in one recording to the similar events in the other recording. It applies typically to songs or other music with a distinct single theme or melody.

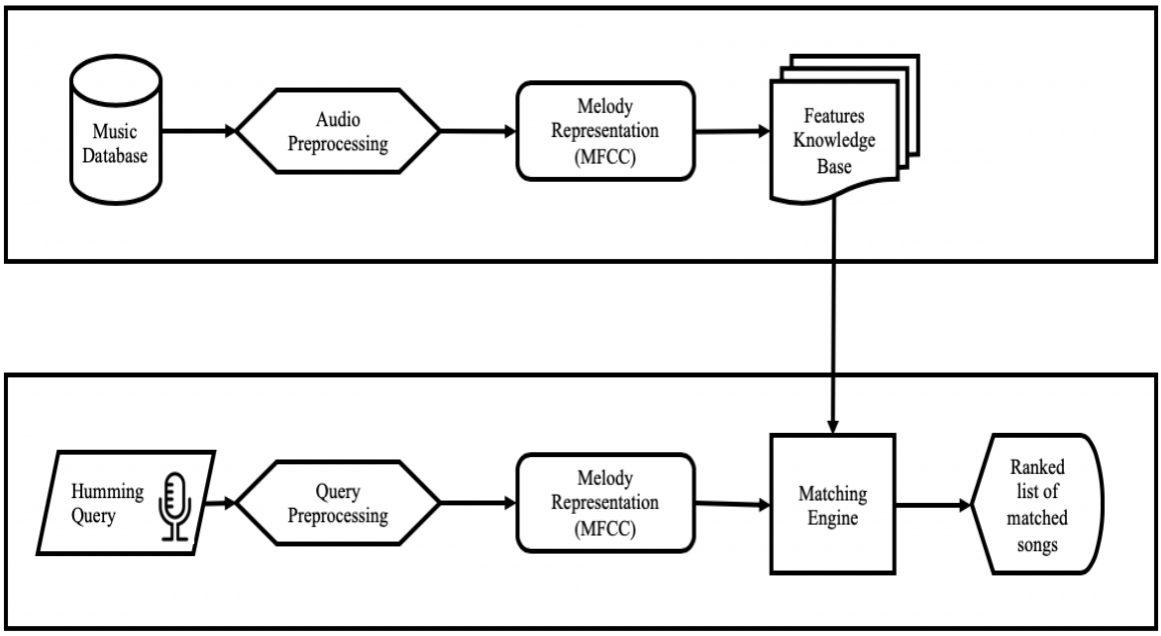

Figure 1. Query by Humming System Flow Chart.

1.2 Definition of Terms

- **A hum** is a sound with a tune in this instance made by producing a wordless tone with the mouth opened or closed, forcing the sound to emerge from the nose. A hum has a particular timbre, usually a monotone.
- **Timbre** is the facet of sound that distinguishes the tone of different instruments and voices even if the sounds have the same pitch and loudness
- **Audio** refers to the production, transmission, or reception of sounds that are audible by humans.
- **An audio signal** is a representation of the sound that represents the fluctuation in air pressure caused by the vibration as a function of time. Unlike sheet music or symbolic representations, audio representations encode everything necessary to reproduce an acoustic realization of a piece of music

- **Symbolic music representations** comprise any score representation with an explicit encoding of notes or other musical events. These include machine-readable data formats such as MIDI. Digital data based on alphabet of letters or symbols are regarded as symbolic.
- The energy of a signal corresponds to the total magnitude of the signal. For audio signals, that roughly corresponds to how loud the signal is.

  The energy in a signal is defined as:

$$\sum_{n} |x(n)|^2$$

1.3 Problem Statement

With the growth of musical data and the availability of cloud-based music streaming access to users, it is essential to innovate on better human-friendly way to retrieve songs from a large dataset.

Music data retrieval in the presence of uncertainty in a vast database is a challenging problem in multimedia information retrieval. In query-by-humming (QBH) systems, uncertainty can arise in query formulation due to user-dependent variability (Unal et al., 2008).

1.4 Purpose of the Research

The goal of the Query by Humming task is to explore MIR system that takes as query audio recording from a user or an audio file hummed by real-world users and find the highest similarity of the input humming.

The objective of this research is to develop an algorithm or a system to retrieve a matching song given a user's query input. To achieve this task, we will be extracting chroma-based statistical features from an audio file for content-based matching. We will explore

experiments and approaches used in a similar study and evaluate the result against our proposed method.

1.5 Organization of the Thesis

The general overview of the chapters in this thesis paper is as follows:

Chapter one introduces the music information retrieval using the dynamic time warping approach. It presents the problem statement and the purpose of the research.

Chapter two reflects on previous related literature. Chapter two projects the general overview of music information retrieval and discusses the effects of scholars and researchers in Query by humming.

Chapter three outlines the methodology of the research. This chapter focuses on data pre-processing and the task of feature extraction. In this chapter, we described the dynamic time warping algorithm which takes advantage of dynamic programming and the longest common subsequence approach to finding the best match for two different pieces of music. It also describes the distance matrix.

Chapter four presents the experiments for the study. This chapter discusses the results of the query using different kinds of features extracted. It also compares the features that perform best for a query by humming project and the varying distance matrix functions to identify which feature performs best for a query by humming project.

Chapter five presents the conclusion and discussion of the research.

CHAPTER 2

Literature Review

The approach to MIR can be categorized based on the information source, e.g., audio signal, symbolic music representation, music metadata or tag, etc. In this research, using Metadata or tag-based approach would separate us from our initial motivation, namely that the computer ‘hears’ a query humming and fetches the best possible outcomes. Hence, we select the raw audio signal as its primary and only source of information

Table 1

Summary of version identification methods and their ways of transcending alteration in musical features

Reference (s)	Extracted feature	Tempo Invariance	Similarity computation
Foote (2000a)	Energy + Spectral	DP	DTW
Yang (2001)	Spectral	DP	Match length
Nagano et al. (2002)	PBFV	Beat + DP	Match length
Izmirli (2005)	Key templates	DP	DTW
Muller et al. (2005)	PCP	Temporal comp./exp	Dot product
Tsai et al. (2005, 2008)	Melody	DP	DTW
Gomez & Herrera (2006)	PCP	DP	DTW
Gomez et al. (2006a)	PCP	DP	DTW
Lee (2006)	Chords	DP	DTW
Marolt (2006)	Melody	DP	Cross-correlation

Table 1

Cont.

Sailer & Dressler (2006)	Melody	Relative	Edit distance
Bello (2007)	Chords	DP	Edit distance
Ellis & Cotton (2007); Ellis & Poliner (2007)	PCP	Beat	Cross-correlation
Kim & Perelstein (2007)	PCP	HMM	MLSS
Ahonen & Lemstrom (2008)	Chords		NCD
Egorov & Linetsky (2008)	PCP	DP	Match length
Jensen et al. (2008a)	PCP	Fourier transform	Frobenius norm
Jensen et al. (2008b)	PCP	2D autocorrelation	Euclidean distance
Kim & Narayanan (2008);	PCP + Delta PCP		Dot product

Table 1

Cont.

Kurth & Muller (2008)	PCP	Temporal comp./exp.	Dot product
Marolt (2008)	Melody	Beat + 2D spectrum	Euclidean distance
Serra et al. (2008b, 2009a)	PCP	DP	Match length
Ahonen (2010)	Chords + Other		NCD
Serra et al. (2010c)	PCP		Prediction error
Di Buccio et al. (2010)	PCP		Set intersection

Source: Serra (2011) Identification of Versions of the Same Musical Composition by Processing Audio Description.

Abbreviations for DP is dynamic programming, HMM is Hidden Markov Models, DTW is Dynamic time warping, PBFV is Polyphonic binary features vector, MLSS is the most likely sequence of states, PCP is Pitch class profile, and NCD Normalized compression distance

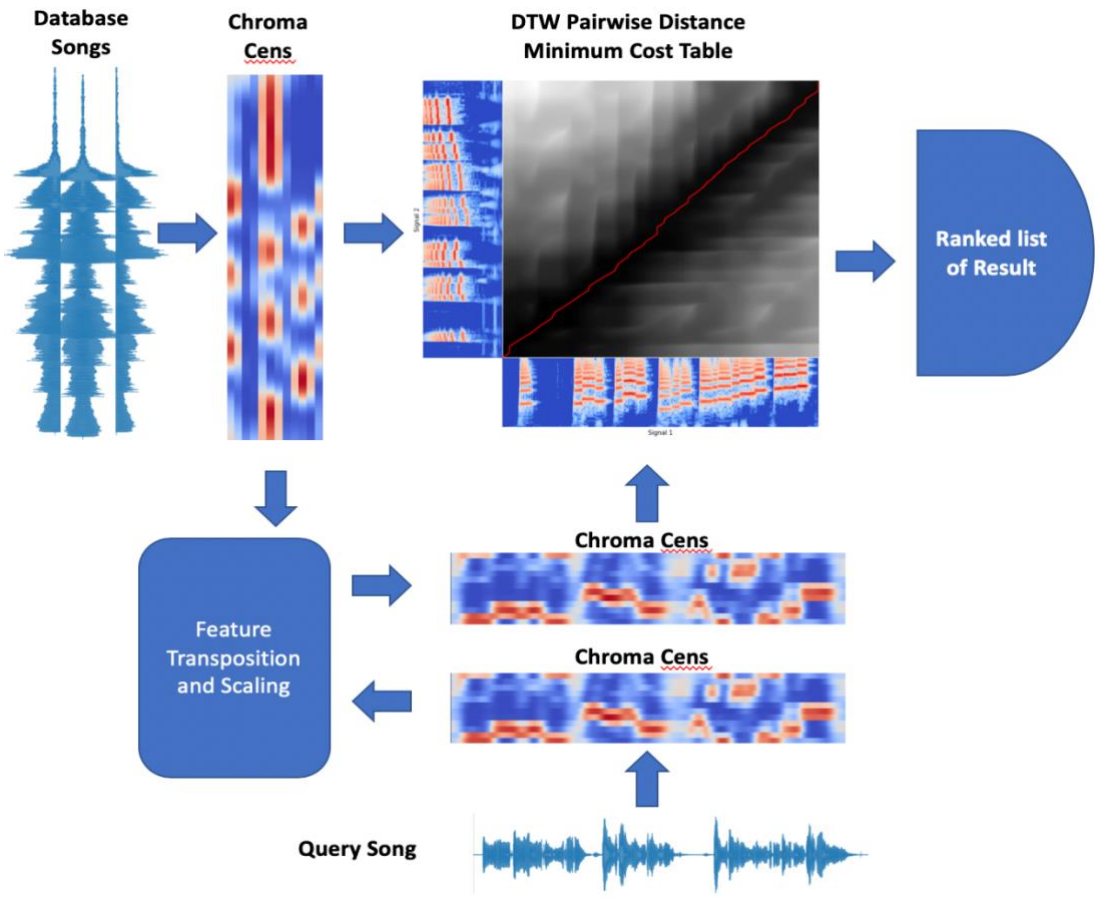

Figure 2.General Diagram of Dynamic Time Warping Approach for Audio-based Retrieval.

Query by humming system architecture explained in Figure 2 consists of the following:

1. Knowledge Base: This contains the dataset of songs and also includes the target song a user will be attempting to retrieve.
2. Query Hum: This is the hum provided by the user via a microphone.
3. Feature Extraction and Preprocessing. For our research we will extract Chroma Constant-Q Transform, Chroma Cens, and Chroma STFT.
4. DTW Algorithm
5. Ranked List of Result.

2.1 Audio-based retrieval

In audio content-based MIR, much effort has been focused on extracting information from the raw audio signal to represent certain musical aspects such as timbre, onset detection, beat tracking and chord estimation. Fingerprinting algorithms have been explored for content-based search using tonal features (Matti Ryyna¨ne, 2008). There have been Experiments to portray the robustness of Query by humming systems using Mel-frequency Cepstral Coefficients (MFCC), Linear Predictive Coefficients (LPC), and Linear Predictive Cepstral Coefficients (LPCC) and the Performance and Precision diminishes gradually with a growing database size (Trisiladevil & Nagappa, 2012). Tree approach has been explored using similarity measures based on statistics derived from a supervised vector quantizer (Foote, 1997).

2.2 Symbol-based retrieval

Most of the research in pre-existing query by humming systems use pitch contour to match similar melodies (Lu, You, & Zhang, 2001). The user's humming is transcribed to a sequence of discrete notes, and the contour information is extracted from the notes. A hierarchical matching technique which is matching pitch contour of higher, lower or similar note has been employed for MIR on symbolic database (Lu, You, & Zhang, 2001). DTW has shown to produce good results on symbolic dataset; (Zhu & Shasha, 2003) addressed using local dynamic time warping on symbolic database.

CHAPTER 3

Methodology

3.1 Data preprocessing

The quality and accuracy of a user hum ultimately affect the result of a query by humming system. If we can extract melody from humming correctly, the desired song can be retrieved more accurately. The objective of the preprocessing algorithm was to strip the leading silence from a signal. We used energy and zero-crossing rate to discriminate silence and noise from a useful humming signal.

3.2 Feature Extraction

Extracting significant feature vectors from an audio signal is a considerable task to produce a better retrieval performance. Chroma-based audio features are robust tool for analyzing music, music synchronization, and audio alignment. A 12-dimensional chroma feature encodes the short-time energy distribution of the underlying music signals over the twelve chroma bands which correspond to the 12 pitch classes in standard Western music: C, C#, D, D#, E, F, F#, G, G#, A, A#, and B.

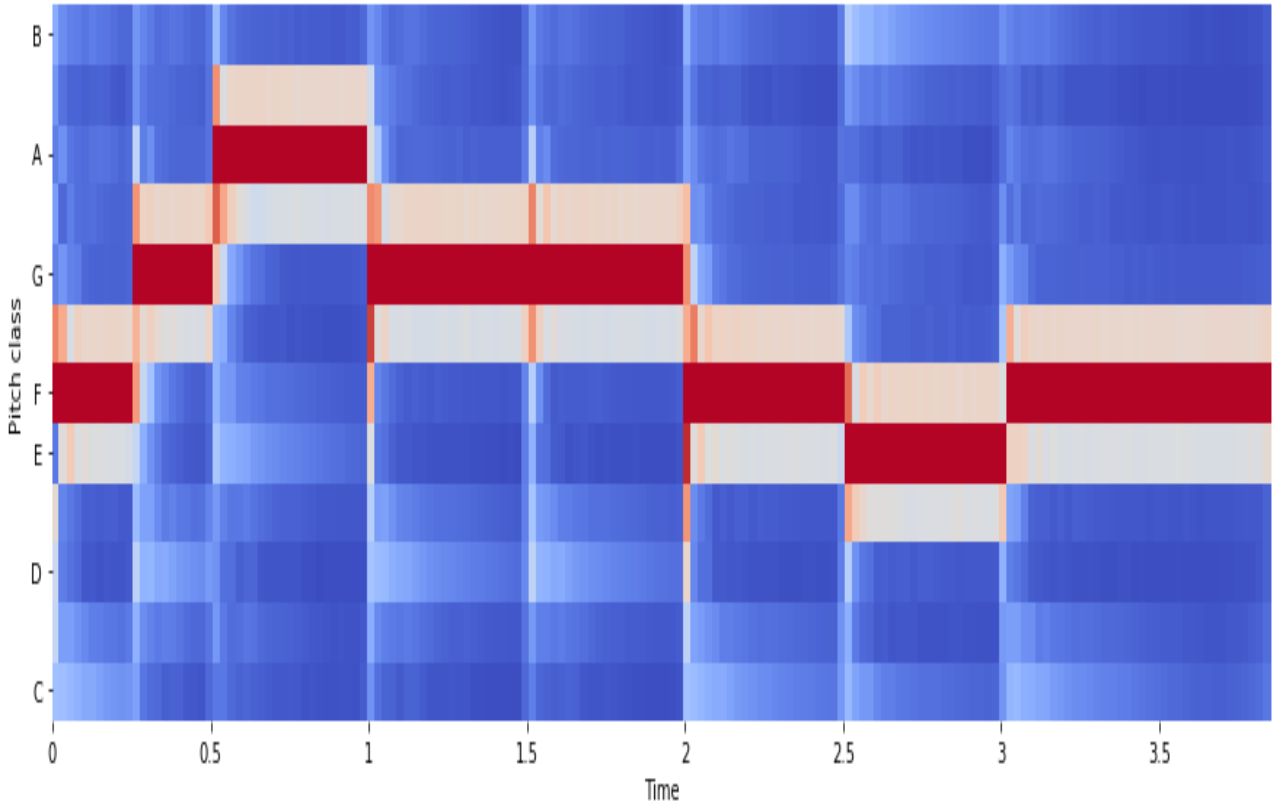

Figure 3. A chroma CQT showing a 12-element feature vector indicating how much energy of each pitch class.

Chroma features capture the harmonic and melodic characteristics of music and very robust to changes in timbre and instrumentation. Performing short-time statistics over energy distributions within the chroma bands results in CENS (**C**hroma **E**nergy **N**ormalized **S**tatistics) features. This smooths the local deviations in tempo, articulation, and musical ornaments.

The main idea of CENS features is that taking statistics over large windows smooths local deviations in tempo, articulation, and musical ornaments. This characteristic makes CENS best used for tasks such as audio matching and similarity.

To calculate CENS:

1. the first step is to decompose the audio signal into 88 frequency bands corresponding to the musical notes A0 to C8.

2. Compute the short-time mean-square power (STMSP) for each of the 88 sub-bands by convolving the squared sub-band signal with a rectangular window corresponding to 200 ms with an overlap of half the size.
3. Add up all corresponding STMPs of all pitches belonging to the respective class to Compute STMSPs of all chroma classes. Then, we compute STM- SPs of all chroma classes C, C#,... ., B by adding up the corresponding STMSPs of all pitches belonging to the respective class. For example, to compute the STMSP of the chroma A, we add up the STMSPs of the pitches A0, A1,... ., A7. This yields for every 100 ms a real 12-dimensional vector $\vec{v} = (v_1, v_2 \ldots, v_{12}) \in R_{12}$, for each analysis window
4. Finally, for each window, we compute the energy distribution relative to the 12 chroma classes by replacing the vectors $\vec{v}$ from Step (3) by $\vec{v} / (\sum_{i=1}^{12} vi)$.
5. Quantize each normalized chroma vector $\vec{v} = (v1, \ldots, v12)$ from Step (4) by assigning the value 4 if a chroma component v_i exceeds the value 0.4 (i.e., if it contains more than 40 percent of the signal's total energy in the ith chroma component for the respective analysis window).
6. Convolve the sequence of the quantized chroma vectors from Step (5) component-wise using a Hann window of length 41. This results in a series of 12-dimensional vectors with non-negative entries, representing a kind of weighted statistics of the energy distribution over a window of 41 consecutive chroma vectors. In the last step, downsample the sequence by a factor of 10 and normalize the vectors to the Euclidean norm.

3.3 Dynamic Time Warping

In MIR, we often want to compare two sequences of different lengths. For example, we

may want to compute a similarity measure between two versions of the same song. We can align two various performances of the same musical work; hence we can hop from one performance to another at any moment in work. This problem is known as music synchronization. DTW is an algorithm used to align two sequences of similar content but possibly different lengths.

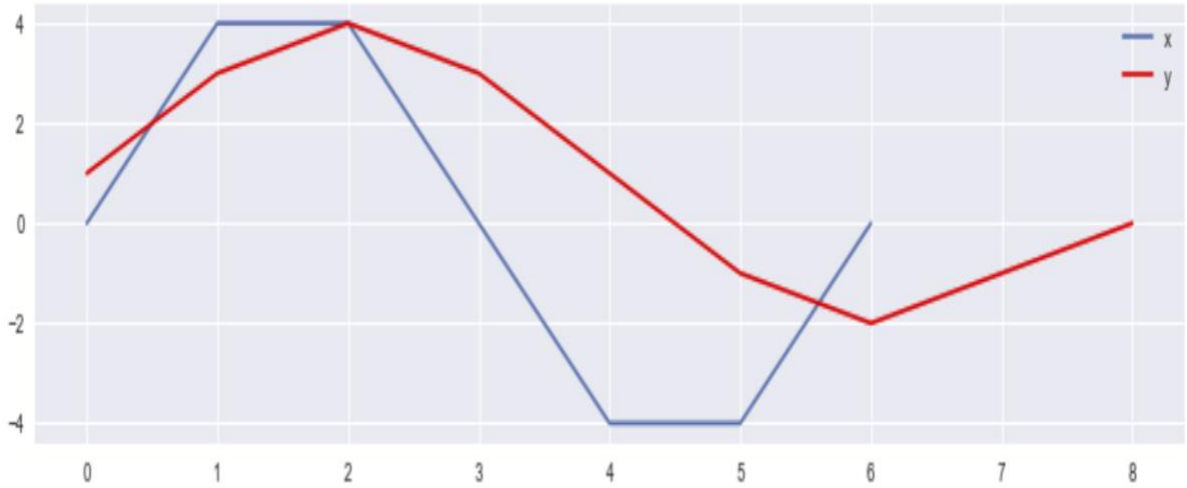

Figure 4. Alignment of arrays. Chroma features are a sequence of vectors.

Dynamic programming string matching best describes the computation DTW distance. The basic idea of DTW is to find a path of index coordinate pairs where the sum of distances along the path P is minimized:

$$min \sum_{(i,j)} d(x[i], y[j])$$

The path constraint is that, at (i, j) (i, j), the valid steps are (i+1,j) (i+1,j), (i,j+1) (i,j+1), and (i+1,j+1) (i+1,j+1). In other words, the alignment always moves forward in time for at least one of the signals. It never goes forward in time for one signal and backward in time for the other signal.

Table 2

Visualization of Optimal Cost Calculation

		1	3	4	3	1	-1	-2	-1	0
	0	inf	inf	inf	Inf	inf	inf	inf	inf	inf
0	inf	1	1	8	11	12	13	15	16	16
4	inf	4	2	2	3	6	11	17	20	20
4	inf	7	3	2	3	6	11	17	22	24
0	inf	8	6	6	5	4	5	7	8	8
-4	inf	13	13	14	12	9	7	7	10	12
-4	inf	18	20	21	19	14	10	9	10	14
0	inf	19	21	24	22	15	11	11	10	10

Here is the optimal substructure. Suppose that the best alignment lies in index pair (i, j), i.e., m_1[i] and m_2[j] are part of the optimal DTW path. we prepend to the optimal path

min {d(m_1[i−1], m_2[j]), d(m_1[i], m_2[j−1]), d(m_1[i−1], j−1])}.

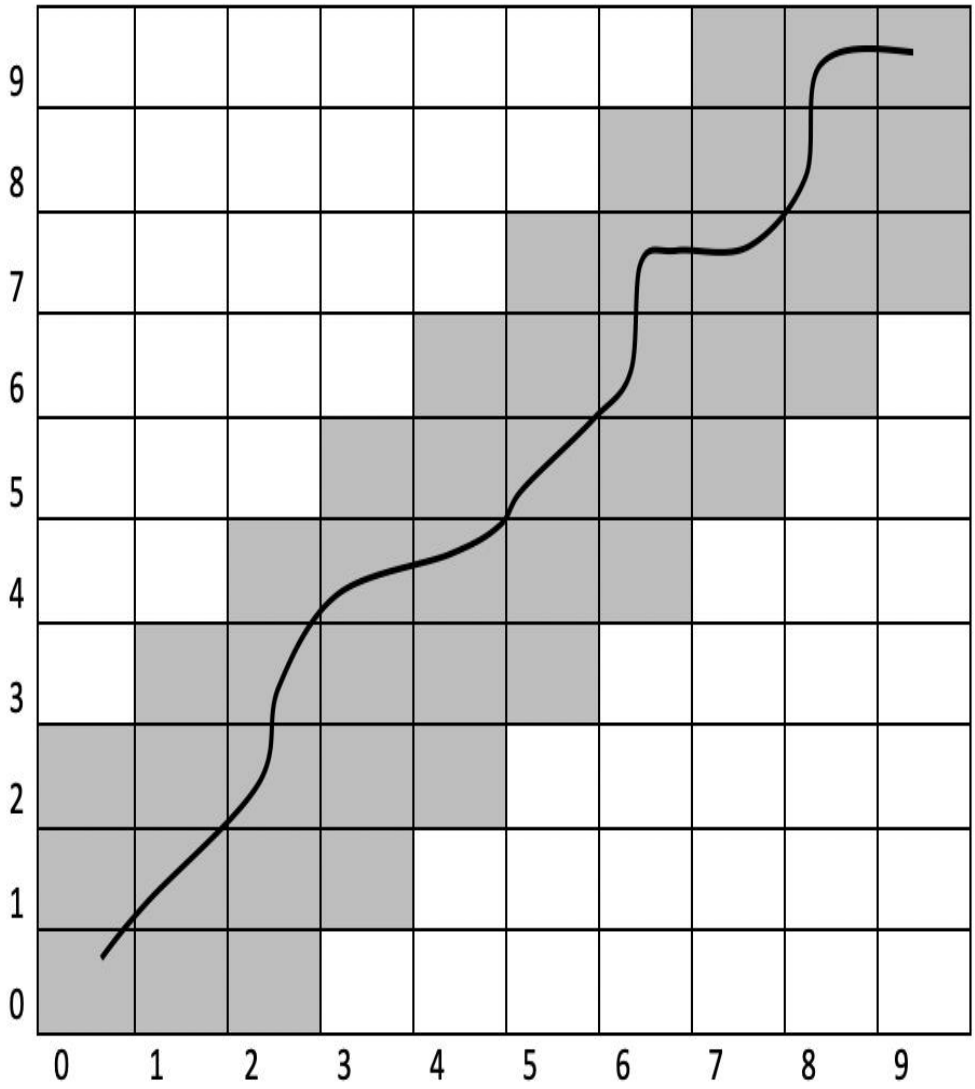

Figure 5. An example of a dynamic time warping path.

3.4 Distance Metric

DTW requires the use of a distance metric between corresponding observations of two musical notes. One common choice is the Euclidean Distance. We would explore Euclidean distance and Manhattan distance for this research.

3.5 Final Ranking

To obtain the final list of retrieved melodies, the candidate melodies are ranked according to their distance to the subset of the query input. The ranking is performed by examining all the matches preserved in the previous step. Each result is stored in a dictionary with the frame and rotation as the key. The results are ranked, and the least score for each unique database song is retrieved.

CHAPTER 4

Experiments

The experiments are divided into three parts. First, we segmented our query input into frames and used a portion of the user humming query as input into our system. We also verified the tune the user is singing in the event the user is singing off-tune. We compared the result of different features and distance metrics.

4.1 Quality of the query by humming system

We collected 50 songs manually and further segmented to frame size of 400. Our query set consists of hum from people with different musical skills. For this experiment, we used the hum queries of better singers in this experiment because for hum queries of poor quality it is hard for even a human being to recognize the target song. We extracted Chroma features from the input query. We tested our system with some hum queries of poor quality against a more quality hum, which performed better.

4.2 Experiments for indexing DTW

We examined the performance of three chroma features: Chroma Cens, Chroma CQT, and Chroma Short Time Fourier Transform. On average, chroma CQT gave the best result for the experiment. A user hum melody is said to be a perfect match if the intended target melody is ranked 1 in the search result.

4.2.1 Query rotation. The index of the query input is rotated to search for the best match with the smallest similarity score. This is important as it always finds the best similarity score for the data, even for a scenario where the input hum is initially off-key.

Table 3

Comparison of Chroma Features Query Hum

Query		**Feature**		
		Chroma_CQT	Chroma_CENS	Chroma_STFT
Query	Target song Title	Target Rank	Target Rank	Target Rank
Hum 1	Mother Nature's Son	1	2	2
Hum 2	Led Zeppelin - Stairway to Heaven	4	2	4
Hum 3	I me mine - The Beatles	11	10	11
Hum 4	Let It Be	1	1	6
Hum 5	Bob Marley - No Women No Cry	5	11	11
Hum 6	Louis Armstrong - What A Wonderful World	3	2	3
Hum 7	Bob Marley - No Women No Cry	7	4	6
Hum 8	The Beatles - Help!	11	4	10
Hum 9	Duke Ellington, Take the A Train	9	7	4

Table 3

Cont.

Hum 10	Imagine - John Lennon	4	4	4
Hum 11	Bob Marley - No Women No Cry	6	10	11
Hum 12	Led Zeppelin - Stairway to Heaven	2	4	1
Average Ranking		5.6	5.1	6.1

Table 3 above shows the target rank of 12 experiments each for Chroma CQT, Chroma Cens, and Chroma STFT. The ranked score compares the performance of the three different features for QBH task.

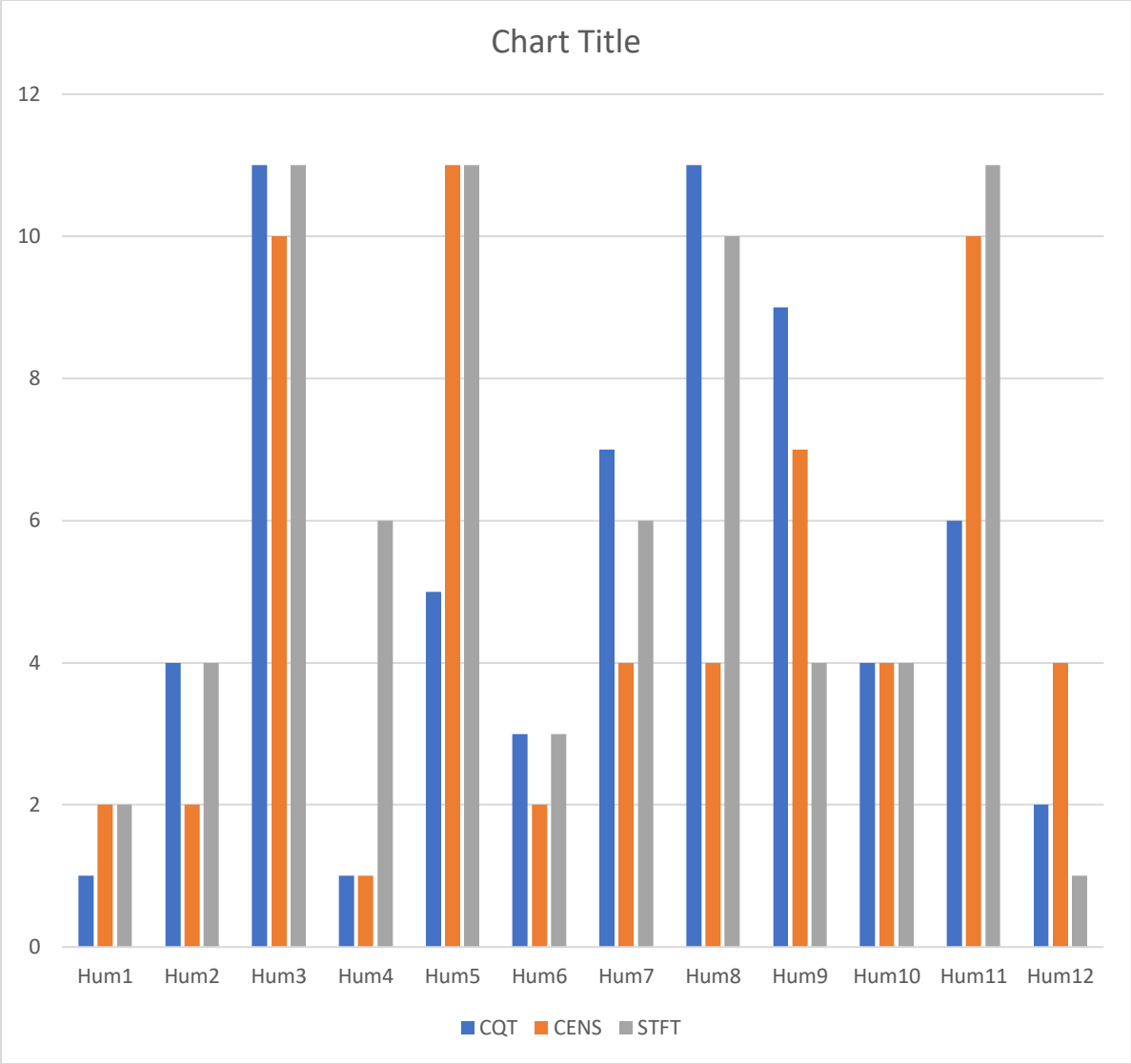

Figure 6. Chroma Features Error Comparison

Table 4

Comparison of Chroma Features Query Hum with knowledge base overlapping

Query		**Feature**		
		Chroma_CQT	Chroma_CENS	Chroma_STFT
Query	Target song Title	Target Rank	Target Rank	Target Rank
Hum 1	Mother Nature's Son	1	4	2
Hum 2	Led Zeppelin - Stairway to Heaven	4	3	4
Hum 3	I me mine - The Beatles	2	4	11
Hum 4	Let It Be	2	1	7
Hum 5	Bob Marley - No Women No Cry	2	11	6
Hum 6	Louis Armstrong - What A Wonderful World	3	4	6
Hum 7	Bob Marley - No Women No Cry	2	7	6
Hum 8	The Beatles - Help!	11	6	7
Hum 9	Duke Ellington, Take the A Train	8	5	1

Table 4

Cont.

Hum 10	Imagine - John Lennon	1	1	5
Hum 11	Bob Marley - No Women No Cry	2	9	10
Hum 12	Led Zeppelin - Stairway to Heaven	2	1	1
Average Ranking		3.3	4.7	5.5

Table 4 above shows the target rank of 12 experiments each for Chroma CQT, Chroma Cens, and Chroma STFT with overlap of the knowledge base.

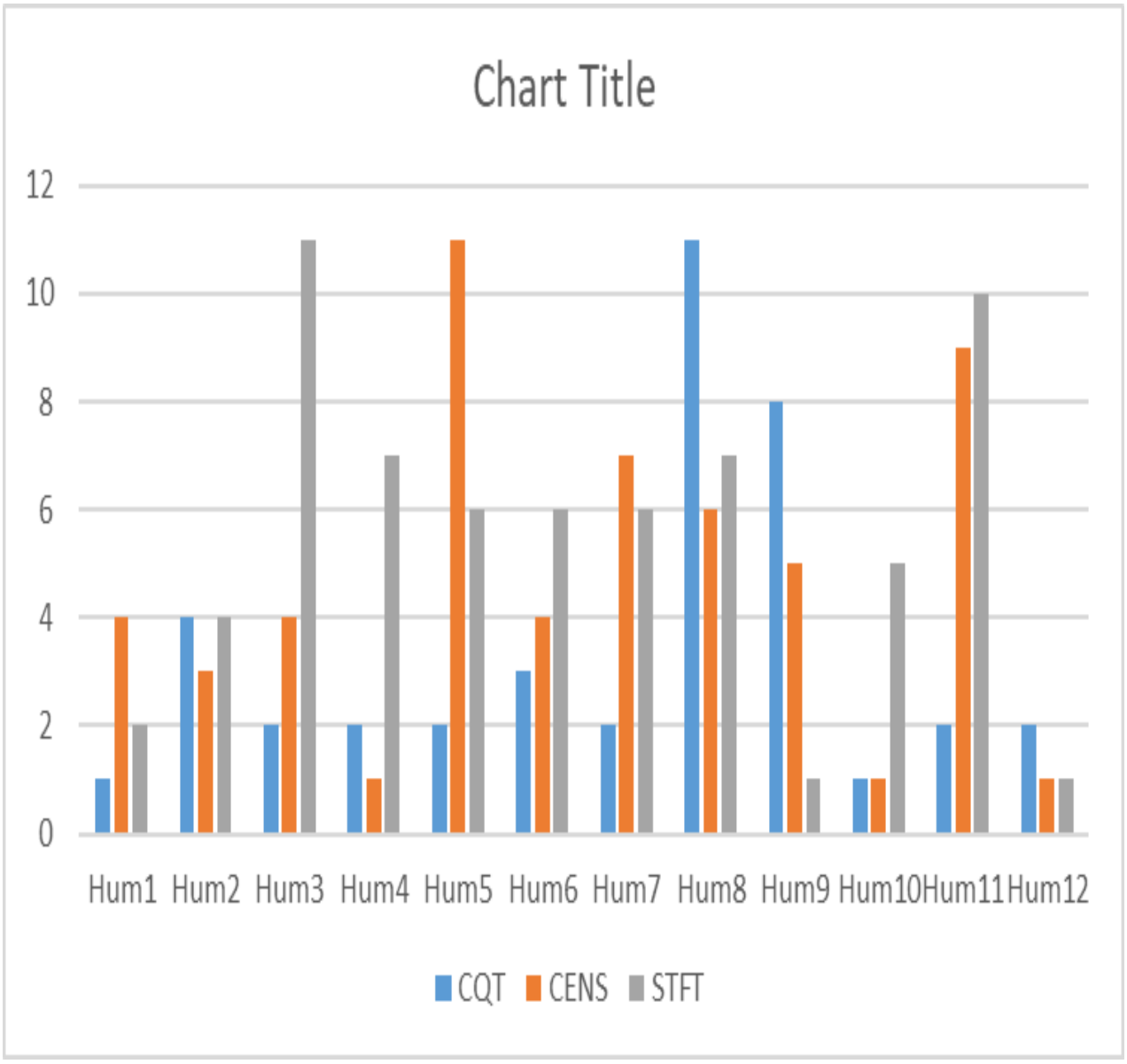

Figure 7. Chroma Features Error Comparison with Overlapping knowledge base.

The ranked score compares the performance of the three different features for the QBH task. The knowledge base was overlapped by one-tenth to improve music alignment. With an overlap, Chroma CQT performed best compared to Chrom Cens and Chroma STFT.

.

Table 5

Comparison of Features on Rotated Query Index

Query		**Feature**		
		Chroma_CQT	Chroma_CENS	Chroma_STFT
Query	Target song Title	Target Rank	Target Rank	Target Rank
Hum 1	Mother Nature's Son	1	5	1
Hum 2	Led Zeppelin - Stairway to Heaven	2	3	4
Hum 3	I me mine - The Beatles	1	1	5
Hum 4	Let It Be	3	3	1
Hum 5	Bob Marley - No Women No Cry	2	1	7
Hum 6	Louis Armstrong - What A Wonderful World	5	3	1
Hum 7	Bob Marley - No Women No Cry	4	5	5
Hum 8	The Beatles - Help!	2	4	5
Hum 9	Duke Ellington, Take the A Train	3	5	7

Table 5

Cont.

Hum 10	Imagine - John Lennon	6	4	4
Hum 11	Bob Marley - No Women No Cry	1	1	4
Hum 12	Led Zeppelin - Stairway to Heaven	3	3	5
Average Ranking		2.8	3.2	4.1

Table 5 above shows the result of 12 experiments to compare the performance of Chroma CQT, Chroma Cens, and Chroma STFT. For each analysis, Query rotation was performed to get the lowest possible similarity for the database files, and a ranked result was produced from the QBH system. We recorded the position of the target song returned from the QBH system and calculated the accuracy of the search.

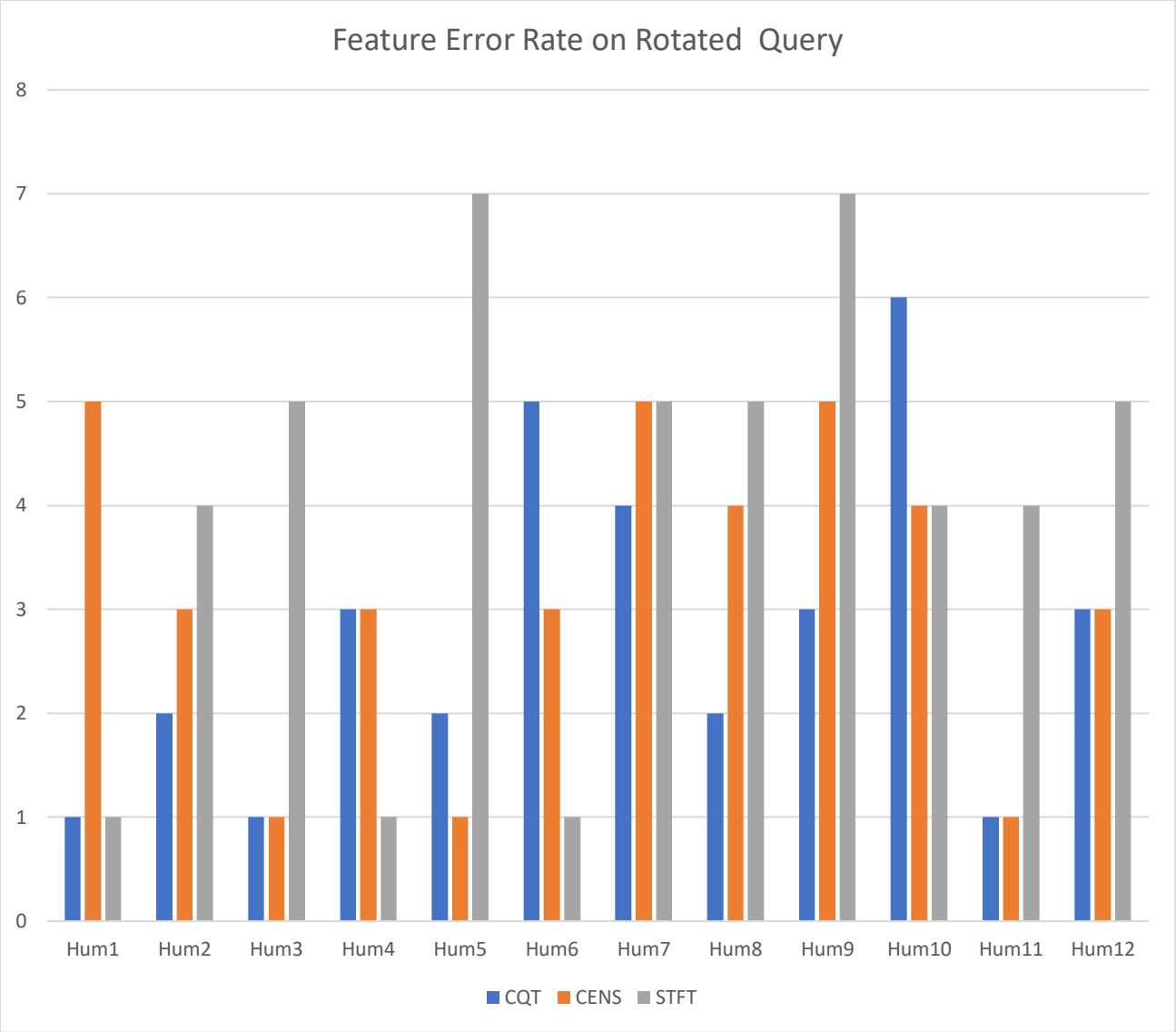

Figure 8. Chroma Features Error Comparison on Rotated Query Index.

A detailed view of each experiment shows the performance of chroma features of the dataset. The error rate of Chroma CQT is consistently lower than on Chroma Cens and STFT. Chroma CQT gave the best performance in 66% of the experiments. Chroma Cens gave the best performance by 41%, while Chroma STFT gave the best performance in 25% of the investigation. The accuracy shows that with query rotation, the result of the search can be

significantly improved. The lowest similarity score is always guaranteed, thereby enhancing the efficiency of the search.

From our experiments, the number of hum melody with perfect match was low. Still, the result is quite encouraging and its possible application. We noticed that changing the portion or frame of user input used as query can significantly alter the result of the search. Also, Increasing the length of the query also produce better humming from poor hummers. We also rotated the users' hum to identity the tune a user is humming to. From our experiments, the similarity value between two music pieces decreases if we rotate the pitch class of the hummed song to find a closer match.

CHAPTER 5

Conclusion

We presented an improved system for retrieving a target song from a time series database using user hum melody as input. Our work compares the different chroma features to identify which is best for a query by humming task. We noticed changing the distance metric from Euclidean distance to Manhattan causes a negligible effect on the result of the search. We compared different Chroma features and Identified Chroma Constant-Q Transform as best among Chroma Cens and Chroma Short Time Fourier Transform for Query by Humming task. We also explore the effect of overlapping extracted database for search. While the impact on the execution time was not significantly different, the result of the search was improved compared to a search without overlapping the knowledge base. Finally, the best performance for our experiment was from rotating the hummed query input to find the best match and overlapping the knowledge base.

Based on our experiment, DTW showed a more promising result for Query by Humming tasks where the input varies from the target as compared to local sensitive hashing. The testing of the system using the query from real people gave good results with high satisfaction. Some improvements will be to implement a dimension reduction algorithm and increase the size of the database for scalability testing.

References

Abas, A. F. (2018). *Euclidian Distance Method for Optimizing Linearly Chirped Fiber Bragg Grating Apodization Profile*. Retrieved 10 21, 2019, from https://scitechnol.com/peer-review/supercontinuum-generation-using-microstructured-optical-fibers-ttsh.php?article_id=7017

Audio. (AUDIO MATCHING VIA CHROMA-BASED STATISTICAL FEATURES).

Birmingham, W. P., O'Malley, K., Dunn, J. W., & Scherle, R. (2003). *V2V: a second variation on query-by-humming*. Retrieved 10 21, 2019, from http://dml.indiana.edu/pdf/jcdl2003demo-web.pdf

Davenport, M. (2012). Introduction to Modern Information Retrieval. 3rd ed. *Journal of The Medical Library Association, 100*(1), 75-75.

Ding, I. J., & Ou, D. C. (2015). Enhancements of SVM Speaker Recognition by Dynamic Time Warping. *Applied Mechanics and Materials*, 891-894.

Dowling, J. W., & Harwood, D. L. (1985). *Music Cognition.* Academic Press.

Duda, A., Nürnberger, A., & Stober, S. (2007). *Towards Query by Singing/Humming on Audio Databases.* Retrieved 10 21, 2019, from http://ismir2007.ismir.net/proceedings/ismir2007_p331_duda.pdf

Foote, J. T. (1997). Content-based retrieval of music and audio. *Published in SPIE Proceedings Vol. 3229:* SPIE.

Guo, Z., Wang, Q., Liu, G., & Guo, J. (2013). A query by humming system based on locality sensitive hashing indexes. *Signal Processing, 93*(8), 2229-2243.

Hawley, M. J. (1993). *Structure out of sound* — Massachusetts Institute of Technology.

Hsu, C. J., Huang, K. S., Yang, C.-B., & Guo, Y. P., (2015). Flexible Dynamic Time Warping for Time Series Classification. *Procedia Computer Science, 51*, 2838-2842.

Kosugi, N., Nagata, H., & Nakanishi, T. (2003). *Query-by-Humming on Internet*. Retrieved 10 21, 2019, from https://link.springer.com/chapter/10.1007/978-3-540-45227-0_58

Kotsifakos, A., Papapetrou, P., Hollmén, J., Gunopulos, D., & Athitsos, V. (2012). *A survey of query-by-humming similarity methods*. Retrieved 10 21, 2019, from http://vlm1.uta.edu/~athitsos/publications/kotsifakos_petra2012.pdf

Li, J., Han, J., Shi, Z., & Li, J. (2010). *An efficient approach to humming transcription for query-by-humming system*. Retrieved 10 21, 2019, from http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.ieee-000005646801

Li, T., Ogihara, M., & Shao, B. (2009). *Machine Learning Approaches for Music Information Retrieval*. Retrieved 10 21, 2019, from https://intechopen.com/books/theory_and_novel_applications_of_machine_learning/machine_learning_approaches_for_music_information_retrieval

Little, D., Raffensperger, D., & Pardo, B. (2007). *A query by humming system that learns from experience*. Retrieved 10 21, 2019, from http://music.cs.northwestern.edu/publications/ismir 2007 v2.pdf

Liu, T., Huang, X., Yang, L., & Zhang, P. (2009). *Query by Humming: Comparing Voices to Voices*. Retrieved 10 21, 2019, from http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.ieee-000005305356

Lu, L., & Seide, F. (2008). *Mobile ringtone search through query by humming*. Retrieved 10 21, 2019, from https://research.microsoft.com/apps/pubs/default.aspx?id=121959

Lu, L., You, H., & Zhang, H.-J. (2001). *A New Approach To Query By Humming In Music Retrieval.* IEEE.

Masood, S., Qureshi, M. P., Shah, M. B., Ashraf, S., Halim, Z., & Abbas, G. (2014). *Dynamic time warping based gesture recognition*. Retrieved 10 21, 2019, from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6828366

Matti Ryyna ̈ne, A. K. (2008). Query By Humming Of Midi And Audio Using Locality Sensitive Hashing. *2008 IEEE International Conference on Acoustics, Speech, and Signal Processing.* Las Vegas: IEEE.

Muller, M., Kurth, F., & Clausen, M. (2005). *Audio Matching Via Chroma-Based Statistical Features.*

Nopthaisong, C., & Hasan, M. (2007). *Automatic Music Classification and Retreival: Experiments with Thai Music Collection*. Retrieved 10 21, 2019, from https://ieeexplore.ieee.org/document/4261369

Phiwma, N., & Sanguansat, P. (2010). *A Novel Method for Query-by-Humming Using Distance Space*. Retrieved 10 21, 2019, from http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.ieee-000005635648

Pourasghar, M., Puig, V., Ocampo-Martinez, C., & Zhang, Q. (2017). Reduced-order interval-observer design for dynamic systems with time-invariant uncertainty. *IFAC-PapersOnLine, 50*(1), 6271-6276.

Raś, Z. W., Zhang, X., & Lewis, R. (2007). *MIRAI: Multi-hierarchical, FS-Tree Based Music Information Retrieval System*. Retrieved 10 21, 2019, from https://link.springer.com/chapter/10.1007/978-3-540-73451-2_10

Rocamora, M., Cancela, P., & Pardo, A. (2014). Query by humming: Automatically building the database from music recordings. *Pattern Recognition Letters, 36*, 272-280.

Sanderson, M., & Croft, W. B. (2012). *The History of Information Retrieval Research*. Retrieved 10 21, 2019, from http://ciir-publications.cs.umass.edu/getpdf.php?id=1066

Serrà, J. (2011). *Identification of Versions of the Same Musical Composition by Processing Audio Descriptions.* Universitat Pompeu Fabra.

Shasha, D., & Zhu, Y. (2004). *Query by Humming*. Retrieved 10 21, 2019, from https://link.springer.com/content/pdf/10.1007/978-1-4757-4046-2_6.pdf

Shen, S., & Chi, M. (2017). *Clustering Student Sequential Trajectories Using Dynamic Time Warping.* Retrieved 10 21, 2019, from http://educationaldatamining.org/edm2017/proc_files/papers/paper_94.pdf

Trisiladevil, N., & Nagappa, B. U. (2012). *An Extensive Analysis of Query by Singing/Humming System Through Query Proportion.* The International Journal of Multimedia & Its Applications (IJMA).

Turning, A. (1950). Computing Machinery and Intelligence.

Unal, E., Chew, E., Georgiou, P. G., & Narayanan, S. (2008). Challenging Uncertainty in Query by Humming Systems: A Fingerprinting Approach. *IEEE Transactions on Audio, Speech, and Language Processing, 16*(2), 359-371.

Wang, Q., Guo, Z., Liu, G., Guo, J., & Lu, Y. (2012). *Query by Humming by Using Locality Sensitive Hashing Based on Combination of Pitch and Note*. Retrieved 10 21, 2019, from http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6266272

Zadeh, L. A. (1984). Making computers think like people. *IEEE Spectrum*.

Zhu, Y., & Shasha, D. (2003). Warping indexes with envelope transforms for query by humming. *MOD International Conference on Management of Data* (pp. 181-192). New York, NY, USA: ACM.

Zhu, Y., Shasha, D., & Zhao, X. (2003). *Query by humming: in action with its technology revealed*. Retrieved 10 21, 2019, from https://nyuscholars.nyu.edu/en/publications/query-by-humming-in-action-with-its-technology-revealed

"Staged Concrete Bridge Deck and Overlay Pours Adjacent [Tender Documents : T24889087]." MENA Report, Albawaba (London) Ltd., Nov. 2014, p. n/a.

Appendix

Thesis Implementation QBH using DTW on Chroma Constant-Q Transform

```
import sys
import os, os.path, pprint, pickle
from collections import defaultdict

import numpy as np, numpy, pandas as pd
from sklearn import preprocessing, decomposition
import scipy.spatial
import random

import IPython.display as ipd
import librosa, librosa.display
# Preprocessing and Feature Extraction
# Database File path
training_dir = 'db/latestdb/'
training_files = [os.path.join(training_dir, f) for f in os.listdir(training_dir) if not
(f.startswith('.'))]
# Parameter
hop_length = 1024
def extractsubfeature(features_dict, filepath):
    filename = os.path.basename(filepath).rstrip('.wav')
    print('ADDED',filename)
    x, sr = librosa.load(filepath)
```

```
    y_h, y_p = librosa.effects.hpss(x)
    C_cqt = librosa.feature.chroma_cqt(y_h, sr=sr, hop_length=hop_length)
    C_cqt = C_cqt.T
    u, count, s = 400, 0, 0
    while s < C_cqt.shape[0]:
        key = filepath +'_'+str(count)
        frame = C_cqt[s:u+s,...]
        features_dict[key] = frame#preprocessing.scale(frame)
        s += u//10
        count += 1
    return features_dict
def getFeatures(training_files, features_dict):
    """A Funtion that extracts Chroma Cqt features from files.
    It returns a dictionary of transposed chroma cqt features.
    """
    print("Added Features: ")
    count = 0
    for filepath in training_files:
        features_items = [f.split('_',1)[0] for f in features_dict]
        if filepath in features_items:
            continue
        features_dict = extractsubfeature(features_dict, filepath)
        if count >= 10:
```

```
            return features_dict
        count +=1
    # remove files in featues not in training folder dictionary
    print("deleting file-features not training folder: ")
    temp = []
    for file in features_dict:
        file_x = file.split('_',1)[0]
        if file_x not in training_files:
            temp.append(file)
        delkey = file.split('_',1)[1]
        if delkey == 0:
            print('REMOVED:', delkey)
            del features_dict[delkey]
    # Deleting files in temp
    if len(temp) > 0:
        for x in temp:
            print('REMOVED:', x)
            del features_dict[x]
    return features_dict
filename_to_load = 'db/latestdb_cqt.pkl'
value = int(input('Enter 1 run create new Feature or 2 to load/update existing features: '))
if value == 1:
    features = dict()
```

```
    features = getFeatures(training_files, features)
    output = open(filename_to_load, 'wb')
    pickle.dump(features, output)
    knowledge_features = features
else:
    features_file = open(filename_to_load, 'rb')
    features = pickle.load(features_file)
    features_file.close()
    # update features from file
    features_update = getFeatures(training_files, features)
    output = open(filename_to_load, 'wb')
    pickle.dump(features_update, output)
    knowledge_features = features_update
print("**knowledge base contains: ", len(knowledge_features), "number of files**")
# Selecting Query File
x, sr_query = librosa.load('db/hum/q44.wav')
x_query, y_p = librosa.effects.hpss(x)
queryFeatureCqt = librosa.feature.chroma_cqt(x_query, sr=sr_query, hop_length=hop_length)
queryFeatureCqt_c = queryFeatureCqt.T
queryFeatureCqt_c = queryFeatureCqt_c[:][100:500]
print('original: ', queryFeatureCqt.shape, 'query: ', queryFeatureCqt_c.shape)
x, x_query, y_p, queryFeatureCqt = None, None, None, None
# Dynamic Time Warping Algorithm
```

```
def dtw_table(x, y, distance=None):
    if distance is None:
        distance = scipy.spatial.distance.euclidean
    nx = len(x)
    ny = len(y)
    table = numpy.zeros((nx+1, ny+1))
    # Compute left column separately, i.e. j=0.
    table[1:, 0] = numpy.inf
    # Compute top row separately, i.e. i=0.
    table[0, 1:] = numpy.inf
    # Fill in the rest.
    for i in range(1, nx+1):
        for j in range(1, ny+1):
            d = distance(x[i-1], y[j-1])
            table[i, j] = d + min(table[i-1, j], table[i, j-1], table[i-1, j-1])
# Execution
%%time
def getResult(features_x, query, sr):
    resultDict = dict()
    y = preprocessing.scale(query)
    for feature in features_x:
        if features_x[feature].shape[0] != 400:
            continue
```

```
        D = dtw_table(features_x[feature], y)
        sim_score = D[D.shape[0]-1][D.shape[1]-1]
        resultDict[feature] = sim_score
    query, y, D, features_x = None, None, None, None
    return resultDict
results = getResult(knowledge_features, queryFeatureCqt_c, sr=22050)
# Result
def printResult(results):
    print("Rank list of result")
    distinct_result = defaultdict()
    position = 1
    for result in sorted(results, key=results.get):
        r_t = result.split('_',-1)[0]
        if r_t not in distinct_result or distinct_result[r_t][0] > results[result]:
            distinct_result[r_t] = [results[result], result.split('_',-1)[1]]
    for filename in distinct_result:
        print("No: ", position,"SongTitle: ", filename, distinct_result[filename])
        position +=1
printResult(results)
```