Fall 12-11-2019

Music Retrieval System Using Query-by-Humming

Parth Patel

Music Retrieval System Using Query-by-Humming

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Parth Patel

December 2019

The Designated Project Committee Approves the Project Titled

Music Retrieval System Using Query-by-Humming

by

Parth Patel

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

December 2019

Dr. Robert Chun	Department of Computer Science
Dr. Thomas Austin	Department of Computer Science
Prof. Kevin Smith	Department of Computer Science

ABSTRACT

Music Retrieval System Using Query-by-Humming

By Parth Patel

Music Information Retrieval (MIR) is a particular research area of great interest because there are various strategies to retrieve music. To retrieve music, it is important to find a similarity between the input query and the matching music. Several solutions have been proposed that are currently being used in the application domain(s) such as Query-by-Example (QBE) which takes a sample of an audio recording playing in the background and retrieves the result. However, there is no efficient approach to solve this problem in a Query-by-Humming (QBH) application. In a Query-by-Humming application, the aim is to retrieve music that is most similar to the hummed query in an efficient manner. In this paper, I shall discuss the different music information retrieval techniques and their system architectures. Moreover, I will discuss the Query-by-Humming approach and its various techniques that allow for a novel method for music retrieval. Lastly, we conclude that the proposed system was effective combined with the MIDI dataset and custom hummed queries that were recorded from a sample of people. Although, the MRR was measured at 0.82 – 0.90 for only 100 songs in the database, the retrieval time was very high. Therefore, improving the retrieval time and Deep Learning approaches are suggested for future work.

Keywords **– Music Information Retrieval, Query-by-Humming, Dynamic Time Warping, Music Instrument Digital Interface, Mean Reciprocal Rank, Similarity Matching**

ACKNOWLEDGEMENTS

I would like to thank my project advisor, Dr. Robert Chun, for his continuous guidance and support throughout this project. Additionally, I would like to thank my committee members; Prof. Kevin Smith and Dr. Thomas Austin for their time as they were available for any support that was required for this project.

Secondly, I would like to thank the San José State University library for providing access to IEEE through which an enormous amount of resources such as articles and research papers were available.

Last, but not least, I would like to thank my family and friends for motivating me throughout the duration of this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Many people use their mobile devices to listen to songs on-demand. People use different methods to search for their favorite song(s) such as search-by-text when users search for a song using a fragment of the lyrics, the artist's name or other means.

Some applications such as 'Shazam' allow users to record a song playing in the background and search its name. However, this method has a drawback which occurs when users do not remember the lyrics to a new song or miss the song playing in the background. A solution to this problem is to use humming as a query to search for songs. Humming refers to emitting a continuous low monotonous sound such as the speech sound when prolonged. This type of system is known as QBH [1, 4, 6, 7]. The proposed system would convert a hummed melody into a piece of music and compare it against the music database to search for the most similar tune/song.

This paper aims to explore the intricacies of QBH systems compared to traditional systems and shed some light on the questions: *What do QBH systems do to address the problem of music retrieval through humming?* and *Does this technique improve accuracy and efficiency over traditional approaches?*

The articles selected for this paper include conference proceedings, published papers and articles related to the field of MIR using QBH.

The following paper is organized as follows: Section 2 presents an overview of the MIR field. It also describes the techniques that are used in MIR systems and their advantages and disadvantages. Section 3 describes the different MIR system architectures

that can be used. Section 4 details the design of the proposed system and its implementation. Finally, Section 5 compares QBH system approaches by mentioning research papers and results. The organization of this research paper will follow the conceptual map as shown in Figure 1.

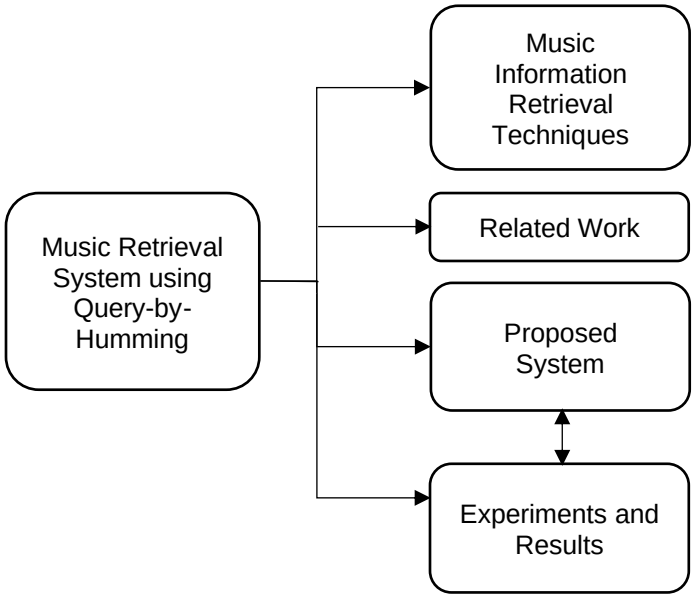

Figure 1: Conceptual map of research paper showing main sections.

2. MUSIC INFORMATION RETRIEVAL TECHNIQUES

MIR is a gradually improving field with a potential future in fast information retrieval. This is because it is very similar to database retrieval; however, MIR uses several different techniques to retrieve music in a fast and efficient manner. MIR spans several different fields such as musicology, psychology, signal processing, machine learning (ML) and optical music recognition. Some applications of MIR are being used by businesses and academics such as recommender systems, automatic music transcription, automatic categorization, and music generation. The remainder of this section reviews techniques used by MIR systems.

2.1. Query-by-Text (QBT)

A QBT technique uses conceptual metadata such as text queries to search for the similarity between a particular song. Applications such as ‘Spotify’ use this feature for their music retrieval system. This was the very first technique that was introduced in the field of retrieval due to its ease-of-use as it relies on previously known text that can be searched through the database.

2.2. Query-by-Example (QBE)

On the other hand, the QBE technique uses a fragment of the original music recording and queries the database to retrieve the most similar song. A famous example of this type of technique that is being used in real-world applications is ‘Shazam’. They use a method known as Audio Fingerprinting [3] to identify or search for an audio based on the

fingerprint created using the query sample. This is a well-known technique that is used currently as it is fast and does not require the full audio sample.

2.3. Query-by-Humming

The QBH technique uses only the natural humming voice emitted from the humans to query the database. Moreover, this approach is suitable as humming occurs naturally and can be attached in the user's mind. A comparison of the approaches in QBH systems is discussed in Section 3.5.

3. RELATED WORK

3.1. Music Information Retrieval System Architectures

MIR system architectures are the structure for the system that is followed in order to retrieve music effectively. They have different architectures because of the different techniques used throughout the system to attempt better music retrieval than other techniques. N. Kosugi et al. [5] indicate that a MIR architecture can use feature vectors [8] that can be extracted from the Musical Instrument Digital Interface (MIDI) file. Similarly, the database will also need to adjust by converting the original song sequences to feature vectors. A similarity search occurs between the two elements and the result is retrieved. In contrast, R. Putri [1] proposes that a time-series data matching algorithm known as Dynamic Time Warping (DTW) can be used rather than the feature vectors. DTW is an algorithm that dynamically measures two different time-series data and produces a similarity. This is an important feature for fields such as speech recognition or MIR because humans have different speaking abilities (e.g. different speed, tone, etc.). Additionally, the time length would differ even if the same person speaks at different times during the day as shown in Figure 2. This method is designed for robustness and can achieve high-retrieval accuracy. Due to the DTW algorithm, the algorithm can adjust to out-of-sync tune/sequence and time warps. Hence, it is more efficient compared to the feature vector approach.

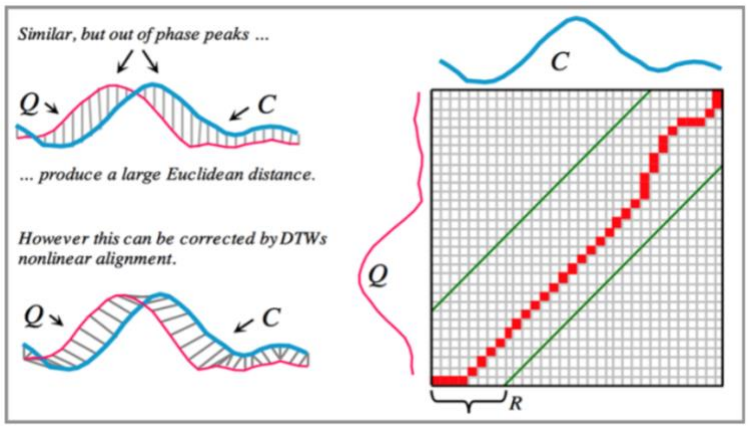

Figure 2: Dynamic Time Warping Theory [13].

3.2. Speech Feature Extraction

Mel Frequency Cepstral Coefficients (MFCCs) are a small set of features that describe the signal's overall shape of a spectral envelope. It is a leading approach for speech feature extraction and has been first introduced to characterize the seismic echoes resulting due to earthquakes. Mel scale is used to measure the perceived frequency of a tone to the actual measured frequency. It scales the frequency to match closely with what the human ear can hear.

3.3. Audio Fingerprinting

Audio Fingerprinting techniques have advanced since the start of music retrieval systems. Until now, there have been several advancements for robust yet efficient algorithm(s) to retrieve music through the use of QBE methods. In 2000, an organization named 'Shazam' released an application [10] that used an algorithm capable of recognizing a short audio sample of music that had been broadcast and mixed with heavy ambient noise. Their algorithm would filter the noise and complete voice codec compression before

reaching their servers. Additionally, the algorithm has to quickly retrieve the music from a large collection of music database with nearly 2M tracks while having a low number of false positives and a high recognition rate. The algorithm only works for audio files present in the database; therefore, it cannot generalize to live recordings even if the artist can sing it perfectly in pitch.

Overall, their audio fingerprinting works by a time-frequency graph called the Spectrogram (Figure 3). This graph is generated for each track, and the algorithm identifies frequencies of peak intensity. For each of the peak points shown in Figure 3, the algorithm keeps track of the frequency at that particular time in the song. Moreover, for a database of around 20 thousand songs, the search time is between 5-500 milliseconds. Their database search over millions of songs has been enhanced to have a look-up time of O(1).

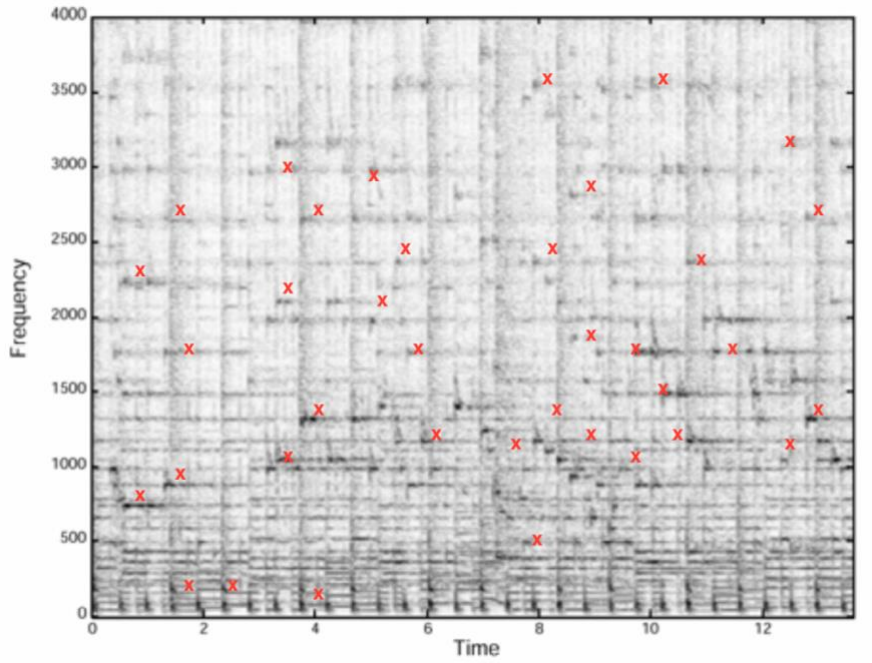

Figure 3: Shazam's generated spectrogram for a particular song marking peak points [10].

On the other hand, T. Jie et al [3] proposes improved algorithms over Shazam to enhance the robustness by using a new audio fingerprinting extraction that employs computer graphics while recognizing the audio samples in complex ambient noise. They also propose a recursive search algorithm based on the confidence measure to improve the retrieval speed. From their results, they conclude that their search strategy is much faster than Shazam's system while matching the accuracy to that of Shazam.

3.4. Machine Learning-Based Music Retrieval

Music retrieval is a wide field with various ways of solving the same problem. ML is an on-going field with continuous research and contributions. ML can be used to train a neural network to accomplish tasks that usually take a long time to process as neural networks are replications of how the human brain processes information. N. Mostafa et al. [12] proposes a Deep Neural Network (DNN) based note-transcription method to train the neural network using hummed notes which are considered as features. These features are passed down the hidden layers of the DNN and these layers help the neural network to train "deeper" from the input query. Using Hidden Markov Models (HMM) combined with Gaussian Mixture Models (GMM) also known as HMM-GMM, their Mean Reciprocal Rank (MRR) accuracy reached to 0.7679 while using DNN-HMM-based acoustic model, their accuracy increased to 0.8071. However, these results are for a small dataset of 4431 songs containing 116 Bollywood artists and they believe the transcription and retrieval accuracy would increase with a larger dataset to train the DNN.

J.-Q. Sun. [13] used a similar approach to train a DNN for a Query by Singing/Humming system (QBSH). Additionally, they compared the Deep Learning (DL) approach with DTW and the results were very similar. For a dataset of size 200, the DTW approach had an MRR of 0.79 while the DL approach had an MRR of 0.82. This portrays that the DL approach is very suitable for better accuracy between the hummed query and the original song; however, they possess a serious problem of gathering the dataset. For a DNN to be effective and provide great accuracy, the dataset has to be massive (in

thousands). Therefore, the DTW approach is not restricted to this problem and can be generalized to a larger dataset using the same system without any major changes.

3.5. Query-By-Humming Systems

Many approaches have been proposed to develop QBH systems such as Contour Extraction [2], HMM [7] and DTW [1, 4]. In 1995, Ghias et al [11] created a QBH system using the contour approach. They used auto-correlation for pitch tracking and convert the melody into a string contour. They use an alphabet of three possible relationships between pitches ('U', 'S' and 'D') representing the situation where a note is above, same as previous note or below, respectively. Furthermore, a string-matching method was used to match between the query and the songs in the database. However, their system was not robust for a large database; hence, the time taken to retrieve the song was increased. In 1999, N. Kosugi [5] proposed a music retrieval system that splits the original music data into sub-data which enables the users to sing/hum any part of the song to retrieve the music. Their system uses both tone distribution and tone transition to enhance the accuracy of the music retrieval. There were various issues that they faced such as shortening the split sub-data and enlarging the music database.

In 2003, Y. Zhu et al. [9] applied DTW and compared the performance between the contour approach and time-series approach. The result showed that time-series data matching had a better retrieval accuracy of 80%. A previous solution described by K. Adamska [2], discusses how the contour approach can be used to retrieve music in a QBH system. The overall idea they propose is to convert the hummed file into MIDI format and

then extract musical contour from it. Musical contour is the estimation of fundamental frequency for each time moment. A MIDI file database would also be converted to a musical contour database. Lastly, the musical data matching algorithm and recognition would recognize the song and retrieve it from the database. The problem with this solution is that it is viable for a small database to get accurate results. However, this creates a challenge for a large music database, and this is normally the case in real-world applications.

In 2005, an organization named 'SoundHound' created a system to retrieve songs based on humming. Their approach is a little different as they use ML to first train the neural network on the hummed songs and then retrieve the song. To achieve this with great accuracy, they have a large database of audio samples that is labeled. They also have another database containing the hummed songs of a random sample of people which are labeled with their original classified song. The model extracts the features of the audio, for example, tone and rhythm. Pairs of humming and the original songs are created and the model computes the features. If the humming and the song are supposed to be the same with their corresponding features, their score increases. The score decreases if they are not similar to each other. Therefore, when a new hummed query occurs, the trained model knows what features it is similar to; hence, it would retrieve the song for you. However, after testing their system, it usually fails in most attempts to retrieve the correct song. For example, sometimes it would show the correct result in their top 5 list or not show the result at all. This problem occurs when the model has over-fitted to the database/dataset and cannot generalize well to the new live data. Furthermore, the live data has ambient noise,

hence, the model would need to know what features to extract without being affected by the noise. Another issue with this type of approach is the time to retrieve the song. It generally takes a long time for music retrieval based on the ML or DL approach due to the computational load of the model.

3.6. Noise Reduction Techniques

Noise reduction techniques are used to remove noise from a particular signal. The field of Natural Language Processing (NLP) continuously use this because it is the main concern. Reducing the ambient noise while the audio input is coming from the microphone can be achieved using well-known techniques. An open-source software known as 'Audacity' can be used for editing audio and reducing the noise. However, for QBH systems, the noise has to be reduced with live input; therefore, data cannot be post-processed using a software. Contrary, for ML approaches in QBH systems, this can be used to 'clean' the data before training the neural network. To understand which noise to cancel, it is important to know the frequency of the overall audio and the humming sound. The method to control or reduce the noise with live input is known as Active Noise Reduction (ANR) which is a method to reduce the unwanted sound by adding a sound specifically designed to cancel out the noise. This would result in a clear audio input that can be further processed. This feature is effective for QBH systems because of the robustness as the female humming frequency range covers up to 350 Hz to 17 kHz while the male frequency range is of 100 Hz to 8 kHz. Therefore, this parameter can be adjusted in the QBH system to check for ambient noise that can be reduced.

4. PROPOSED SYSTEM

4.1. Dataset

In this section, we briefly describe the dataset used for this particular system and how this dataset was transformed to fit the needs of this system. The dataset format was MIDI which was downloaded from 'MIDI Dataset' [14] that contained 77,153 number of songs. A comparison of file formats according to their size is shown in Figure 4. Although, the MP3 file format takes up less disk space, comparative research was done to select the MIDI file format for this project (as shown in Table 1). The MIDI file format also reduces storage space in comparison to WAV or MP3 file formats. An initial dataset was prepared with a sample size of 5 different English songs that also had different genres. For testing purposes, it was essential to start from the basics and gradually improve the system to fit for 77,153 songs.

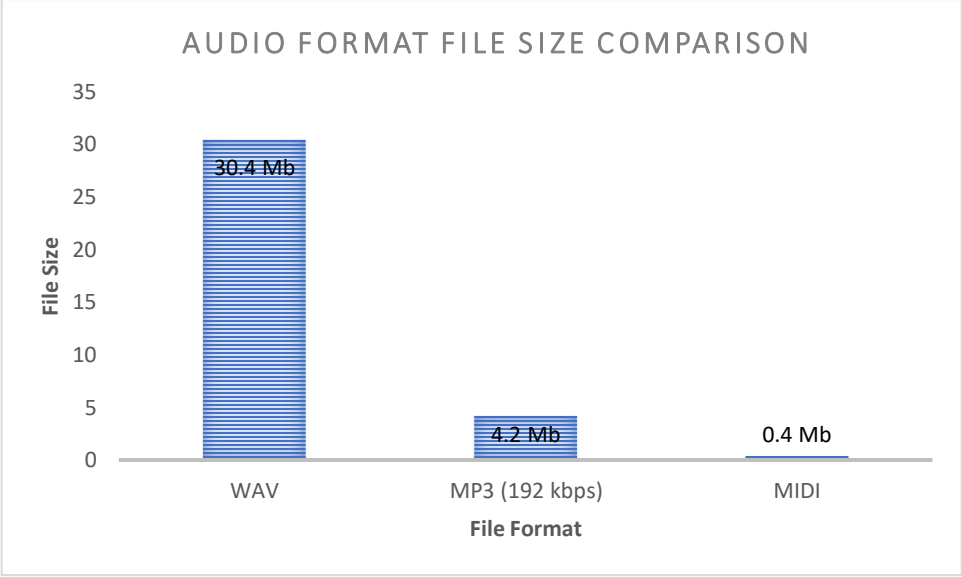

Figure 4: Best audio file format to occupy less disk space [16].

	Typical MIDI file for a song	Typical WAV file for a song
File size	Small (0.2 – 0.6 Mb)	Very large (20 – 60 Mb)
Data extraction	Allows more useful musical data to be extracted.	Contains all the data to be extracted; hence, increasing file size.
Loading time	Short (around 20 ms)	Long due to the large file size. (over 500 ms)
Ease of editing	Allows more precise editing	More difficult to edit

Table 1: Comparison of MIDI and WAV file formats.

4.2. Technical Stack

The following libraries and frameworks were used in the development of this system:

Component	Library/Framework
Programming Language(s)	Python
Database(s)	SQLite
Time-series analysis	Dynamic Time Warping
Data processing	Matplotlib, Scikit-learn metrics, Numpy

Table 2: The technical stack used for the proposed solution.

4.3. Data Preprocessing

To process this data further, the MIDI format was used because it holds a variety of song information that can be extracted such as pitch, contour, instrument type, etc. Therefore, pitch was selected to be a best-fit feature for this system; hence, pitch vectors were extracted from all the MIDI files.

4.4. Database Preparation and Storage

Following the extraction of pitch vectors, this data was stored onto a local SQLite database. A table named “songs” was created to hold the following song information as columns:

1) Song ID
2) Song Name
3) Song’s Pitch Vector

This database was created to enhance the efficiency of the song search and retrieval while reducing the storage space in the system.

4.5. Music Retrieval System

In the proposed approach, we use humming as the input query and parse it to the humming transcription module. This module essentially converts the query into a MIDI file and extracts important features from it such as Pitch since this is the best feature to extract for checking similarity on humming and original audio. Feature vectors can be extracted from these features that show a representation of the feature.

The retrieval module would perform DTW using the feature vectors and find a similarity search between the hummed query and original songs from the database. The database would also hold the feature (pitch) vectors of the original songs. The similarity search would then retrieve the top *k* results (ranking in order of best to worst) and output

this to the user. For a closer and accurate match, the result would only output as a single song rather than retrieve the top *k* results.

Overall, the main difference we use in the proposed solution compared to previous ones is to use feature vectors for the DTW algorithm and to change the important features to the pitch of the music. To allow for faster retrieval and reduced memory footprint, the system compares the feature vectors from live input and the database using the DTW algorithm.

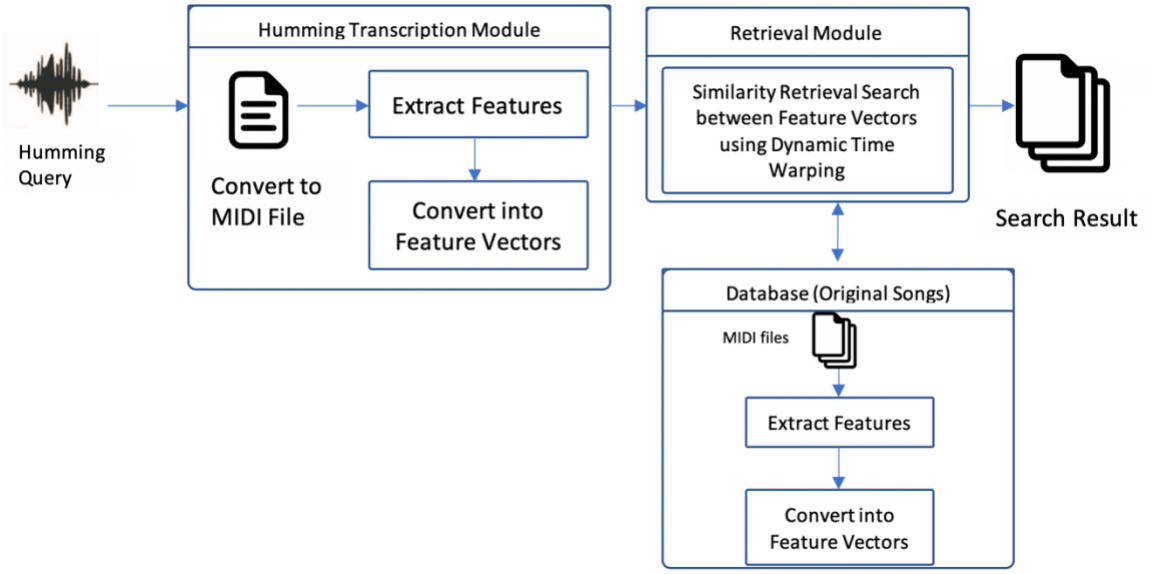

Figure 5: System architecture of the proposed solution.

The system is evaluated using the index known as Mean Reciprocal Rank (MRR):

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

where N is the number of queries and r_i refers to the rank of the correct answer in the retrieved songs for the i-th query.

4.5.1. Proposed solution with Database querying

After finding the results of the previous method, this approach was sought to be more useful due to its use of database and the live query input. The humming transcription module converts the input hum to a MIDI file and extracts the pitch vectors of it. Additionally, a script creates an SQLite database from a database directory consisting of multiple songs. The creation of this database results in fast database search and retrieval and less storage space. In retrieval, the pitch vectors of the hummed query can be compared against the pitch vectors in the database using the DTW algorithm. Since we are comparing two audio features, the main problem in MIR is **music synchronization.** For example, we may want to compute similarity measures for variations of the same song; however, there could be timing deviations. This is where DTW helps solve the problem by aligning the two sequences by factoring the timing and length deviations. For example, consider the given two sequences, $x[n], n \in \{0, \dots, N_x - 1\}$ and $y[n], n \in \{0, \dots, N_y - 1\}$. DTW computes the similarity and produces a set of index coordinate pairs $\{(i, j), \dots\}$ such that $x[i]$ and $y[j]$ are similar [15]. The final retrieval is sorted (by most similar song to least) according to the cost of the distance cost calculated from the DTW. However, if there is a strong match in comparison to the other songs in the database, it would output the top match. The time complexity of the DTW algorithm is *O(X.Y)* where *X* and *Y* are the two pitch vectors.

The below are some screenshots displaying a user scenario of the proposed QBH system. For example, Figure 6 shows the Python scripts collecting pitch vectors for songs from a directory of MIDI files.

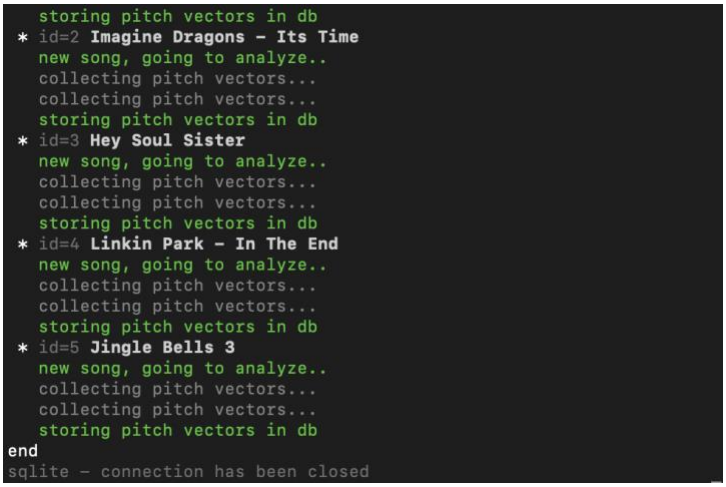

Figure 6: The proposed system collects pitch vectors and stores into the database.

A second script was developed and used to record voice from the microphone where the visualizer console library was imported to visualize the peaks of the microphone recording (as shown in Figure 7). The user would specify a 'seconds' parameter and the microphone would be open until that time. After finishing, the system would measure similarity from the query pitch vector against the database and it would display the song name that closely matches the query. However, by doing a debug of the system, we were able to manually measure the MRR of every query from the sample by printing songs in order of similarity as shown in Figure 8. To automatically measure MRR, there needs to be a ground truth containing hummed queries of every song. However, ground truth dataset was not available with the MIDI dataset; therefore, we had to manually calculate the MRR of every query. A final output of the system can be seen at this YouTube video link: https://www.youtube.com/watch?v=l_pl51h2PMc

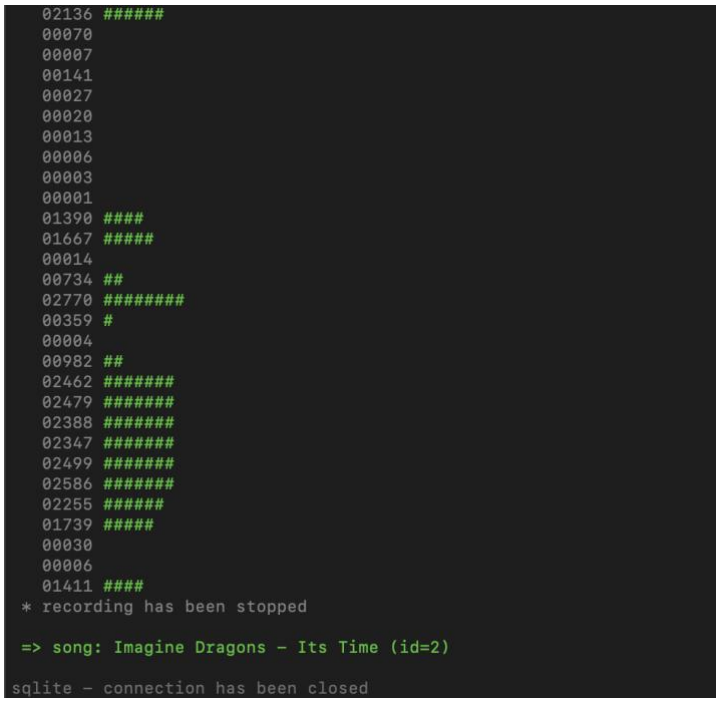

Figure 7: A hum is used as a query and the correct song has been identified.

```
['Imagine Dragons - Its Time',
 'Jingle Bells 3',
 'Linkin Park - In The End',
 'Hey Soul Sister',
 'A Thousand Years']
```

Figure 8: The result output displayed in debug mode.

5. EXPERIMENTS AND RESULTS

During the experimental stage of the project, we wanted to do several tests on the system to measure metrics such as music retrieval accuracy, information search and retrieval speed and performance against other approaches. This section of the report depicts the types of experiments that occurred to evaluate the system against multiple methods and factors.

5.1. Evaluation of database size

The MIDI dataset consists of 77,153 songs; hence, in order to test that, the system has to gradually start from a small amount of data. Therefore, the first test sample was of size 5. The reason why the SQLite database was chosen to store the song information is because of its low storage capacity compared to other file formats. The size comparison evaluation is portrayed in Table 3. A file format storage comparison graph (as shown in Figure 9) displays how the WAV file format deviates drastically in size which it reaches to over 2TB in size for the full dataset. In comparison, MP3 is second highest at 753GB, while MIDI is at approximately 17GB. The research done by R. A. Putri and D. P. Lestari [1] consisted of MIDI files; however, as shown in Table 3, SQLite database storage takes up less disk space compared to MIDI files. An SQLite database is memory efficient and fast in information retrieval when querying for the correct songs. As the database has to be searched for all the pitch vectors that contain the closest value in association to the query's pitch vector, and retrieve the song name(s), the storage has to be kept at a minimum to achieve this task with faster speed.

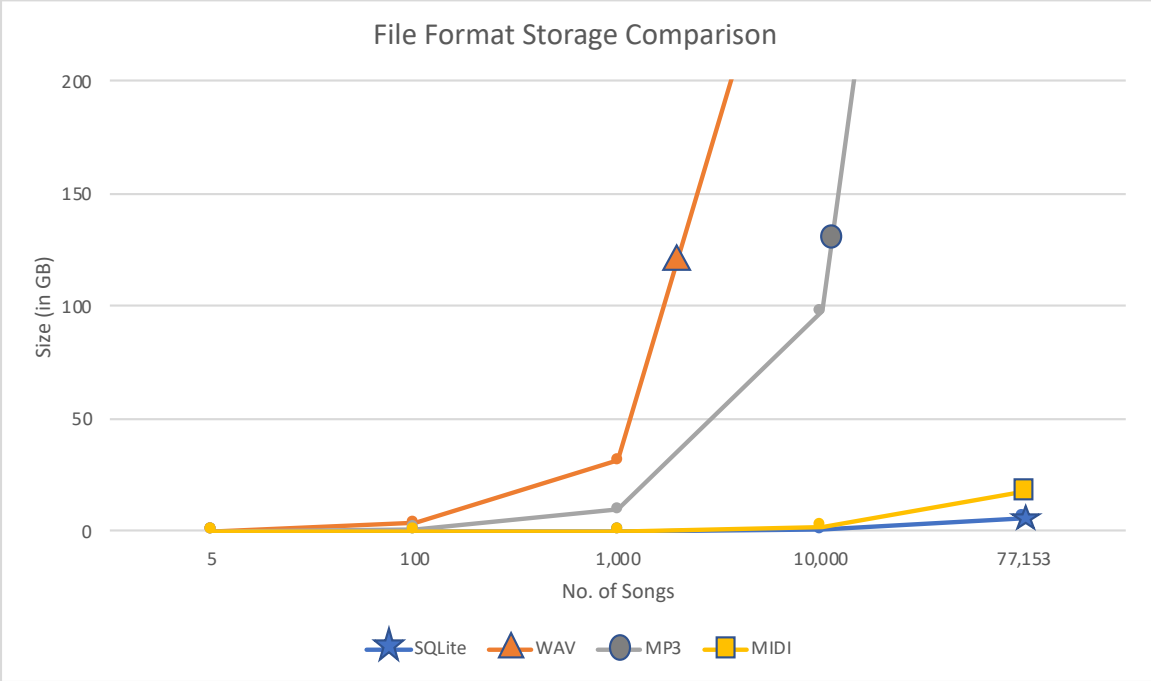

Figure 9: A size comparison graph based on the file formats.

No. of Songs	File Formats Size (in Gigabytes)			
	SQLite	**WAV**	**MP3**	**MIDI**
5	0.0004	0.2483	0.0488	0.00112
100	0.0072	3.4254	0.9821	0.0254
1,000	0.0794	31.4526	9.7322	0.2323
10,000	0.7843	336.7228	97.5906	2.2377
77,153	6.16725	2326.1603	753.0136	17.2903

Table 3: A file format table depicting the sizes against a number of songs.

5.2. Initial experiment - MFCC

In the initial phase of the project, we used a method known as Mel Frequency Cepstral Coefficients (MFCCs) which are a small set of features that describe the signal's overall shape of a spectral envelope. It is a leading approach for speech feature extraction;

therefore, we decided to move forward to enhance this MIR system. The initial dataset of 5 songs was chosen for this approach and the code was written using the Python programming language. Due to this being the first approach, we used a hummed query file to compare against these 5 songs rather than a live query input from a microphone. The implementation consisted of the following steps:

1. Load the input query beforehand by providing a relative path to it.
2. Calculate the MFCC for the hummed query and extract the features.
3. Iterate through the song database, load each file, calculate the MFCC and extract the features for each song.
4. Compare the MFCC of the hummed query and each song using the DTW algorithm.
5. Compute the cosine similarity using the extracted features.
6. Finally, sort the retrieved results using cosine similarity from most match to least.

5.3. Evaluation of MFCC and proposed solution

The result of the first experiment that compares the MFCC approach and the proposed solution is shown in Table 4. The experiment was to query the system by providing a snippet of the hummed tune that can be used to compare against the number of songs in the database. Firstly, the song sample was 5, which was then increased to 100. At 5 songs, the MRR for the MFCC approach resulted in 0.54 which depicts that this method is a good start; however, the retrieval time was on average 1 minute. This is because it would open each audio file and measure similarity. This is a huge overhead on the system even though the accuracy is viable. Moreover, we tested the system's scalability by adding more songs and the result was such that, for 50 queries, the MRR was 0.71. This is a big

increase in the ranking of the songs. It depicts that for 50 queries, the correct song was always in either first, second or third rank out of all the songs. There was a false positive rate of 0.2 meaning that some songs were similar to the hummed tune even though they aren't the correct song. For 5 songs, the proposed system had a false positive rate of 0.17 while for 100 songs, it was 0.2. In comparison, for 5 songs, the MFCC approach had a false positive rate of 0.3 whereas for 100 songs, it was 0.32. The reason for this was the variability in genre that was found between 'Indie' and 'Pop rock' songs which has similar tunes across many songs.

The proposed QBH system performed reasonably well because the key component was the database that would store only the pitch vectors of each song. The MRR of the proposed system for 5 songs was 0.82 compared to the MFCC approach which was only 0.54. This portrays that the proposed QBH approach is more accurate than the MFCC approach. Humming is similar to the pitch vector because lyrics/words are not a factor. The only requirement is the song's tune which is captured in the MIDI file.

Overall, the MFCC approach had to be abandoned because of its slow retrieval speed and the MRR values for the proposed QBH system was better in comparison.

Method approaches	**Mean Reciprocal Rank (MRR)**	
	5 songs	**100 songs**
MFCC	0.54	0.71
Proposed QBH System	0.82	0.90

Table 4: MRR of the MFCC and Proposed QBH approach according to a number of songs.

5.4. Evaluation of related approaches

To evaluate the proposed system better, we compared it against related approaches. Several of these were different MIR techniques as discussed previously; therefore, it was difficult to match the scalability of some commercial products.

5.4.1. Audio-Fingerprinting/Shazam

It can be deduced that audio fingerprinting solutions such as Shazam do have some false positives when presented with noisy queries as it can be confused with different types of songs that have a similar tune. A major problem with Shazam is that it can only be used for music playing in the background; hence, humming is not an option. Additionally, it will not work for original songs that are sung differently by various YouTube artists. Although, it does provide fast retrieval speed for example, it took a search time of 5-500 milliseconds for about 20,000 songs in the database [10]. Overall, Shazam provides a low number of false positives, a high recognition rate, and a faster retrieval speed.

Comparatively, the proposed system is a QBH approach; therefore, its evaluation against audio-fingerprinting provides biased results. The retrieval speed for the QBH system for 20,000 songs was measured to be in the order of 3-6 seconds. Although this is slower compared to Shazam's retrieval speed, it was not the worst for a QBH system that did not have noise resistance and still provided an MRR ranging between 0.85 - 0.90.

5.4.2. Deep Learning

The research evaluation carried out by J.-Q. Sun and S.-P. Lee [13] suggests that Deep Learning methods show better performance and accuracy compared to the DTW approach. Their DL system on a size of 200 songs produces an MRR of 0.82. However,

the proposed system has already achieved 0.90 MRR at 100 songs in the database. This is a major improvement because the proposed system can achieve at least 0.90 MRR for the duration of the tests on different sample sizes. The DL has a huge factor in collecting the dataset used for training/testing the neural network. J.-Q. Sun et al. collected 10 songs of hum each from 10 different people where the environment was quiet. Therefore, the DL method is less effective when noise is introduced in the input hum query. Similarly, people have different voice tone; therefore, collecting hummed tunes from just a sample of 10 people seems less useful because in a new given scenario, one could have a voice that is different from the trained dataset. This would result in several false positives and false negatives.

Another research conducted by N. Mostafa et al. [12] suggests that Deep Neural Network-based note-transcription is a good technique that got encouraging results. Their HMM-GMM (Hidden Markov Model – Gaussian Mixture Model) based acoustic model achieved an MRR of 0.7679 while the DNN-HMM-based acoustic model achieved 0.8071. This is a different technique to DTW as it uses pitches as the notes from the query are transcribed and matched against the notes in the songs using string matching techniques. However, this technique still has room for improvement as it has only been tested to work on a small dataset. The problem with DL methods is that they require time to train the neural network, computational power, and the training data.

5.4.3. SoundHound

SoundHound is an application that uses humming to find the matching song stored in the database. However, due to no research paper available, there is no way to evaluate it

against the proposed system. The application is available for anyone; hence, we tested it across 100 hummed queries and the MRR was around 0.73. Moreover, the application produced a false positive rate of 0.6. An increase in the false positive rate occurs when the model has over-fitted to the database/dataset and cannot generalize well to the new live data. Secondly, the live data may involve noise and the ML model cannot distinguish between the song behind the noise. Therefore, using A.I. is only good for QBH systems if there is a large training data available while adding noise reduction techniques that can be used to prevent the model from producing false positives and false negatives.

5.5. Evaluation of gender-hummed queries

In QBH systems, many factors cause the variability in MRR values for example, the differences between male or female voices and whether the hummed tune is from a professional or non-professional singer. We evaluated the system based on these factors and the results are discussed as follows. A sample of 3 males and 3 females who were close friends were chosen to query the system to record the MRR.

As shown in Figure 10, the frequency of male and female voice is different between various voice categories. This could result in a change in MRR in the proposed system; therefore, the result of this evaluation is shown in Table 5. The MRR varied slightly for male and female voices as expected. Due to the difference in pitch and frequency, the similarity shows different songs and at a different ranking. As a result, the MRR value drops from the general result. The mean value for male queries against a sample of 5 songs was recorded at 0.663 where female queries were at 0.66. For a sample of 100 songs, the mean of male queries was 0.84 whereas female queries had 0.80. This displays the change

in MRR for different types of people according to their vocal abilities. The experiment that we conducted consisted of a random sample of 6 people that had either high-pitched or low-pitched voice. These people were later divided into non-professional and professional singers. The average of the male and female MRR from the random sample of 6 people is shown in Table 5.

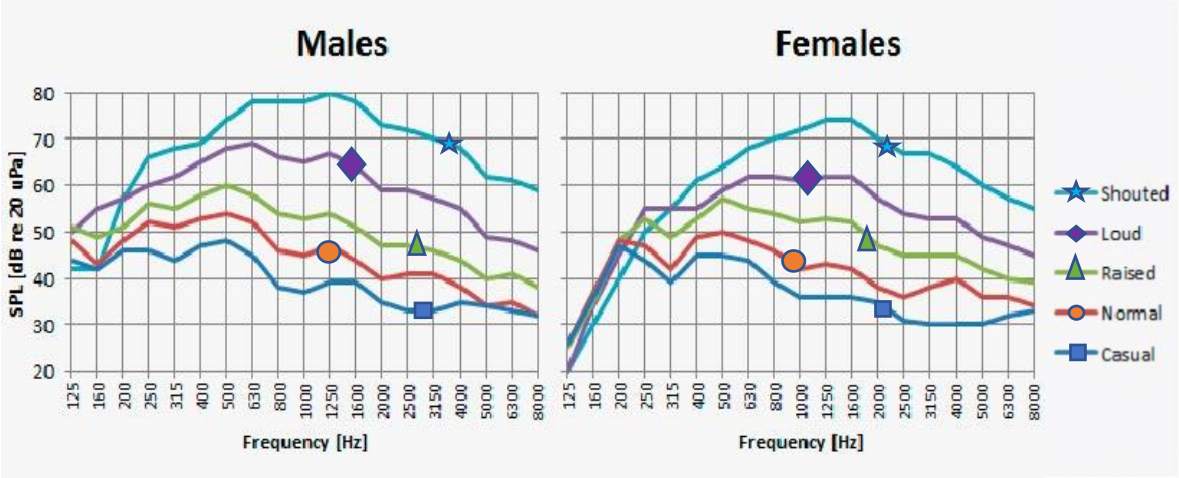

Figure 10: The frequency (Hz) of Males and Females across different voice categories [17].

Gender	Mean Reciprocal Rank (MRR)	
	5 songs	100 songs
Male	0.663	0.84
Female	0.66	0.80

Table 5: Average MRR of gender-based queries against different song samples.

5.5.1. Professional and Non-professional singers

Humming in-tune makes a major difference in evaluation results in any QBH system. This is a common factor because in singing, not everyone can do it in-tune. Professional singers can sing/hum in rhythm; therefore, it would be a very close match to

the song when measuring the similarity compared to non-professional singers. The results for these are shown in Table 6. For this experiment, this project recruited 2 singers from the total sample of 6, who were either professional or non-professional at singing. This resulted in very different MRR when matching their queries with the songs in the database.

Type of Singers	**Mean Reciprocal Rank (MRR)**	
	5 songs	**100 songs**
Professional	0.82	0.90
Non-professional	0.64	0.72

Table 6: MRR of the types of singers against different song samples.

5.6. Evaluation of Performance

The two main factors for the proposed system are MRR and database retrieval speed. In the above sections, we evaluated the MRR; hence, database retrieval speed is presented in this section. The database speed (as shown in Figure 11) was measured at around 2.2 seconds for 5 songs. However, as the number of songs increases in the database, the time increases. The query matching is measuring similarity across every pitch vector for all songs; therefore, the retrieval time is large. I discuss some ideas on improving the algorithm to retrieve the song in the ‘Future Work’ section of this project.

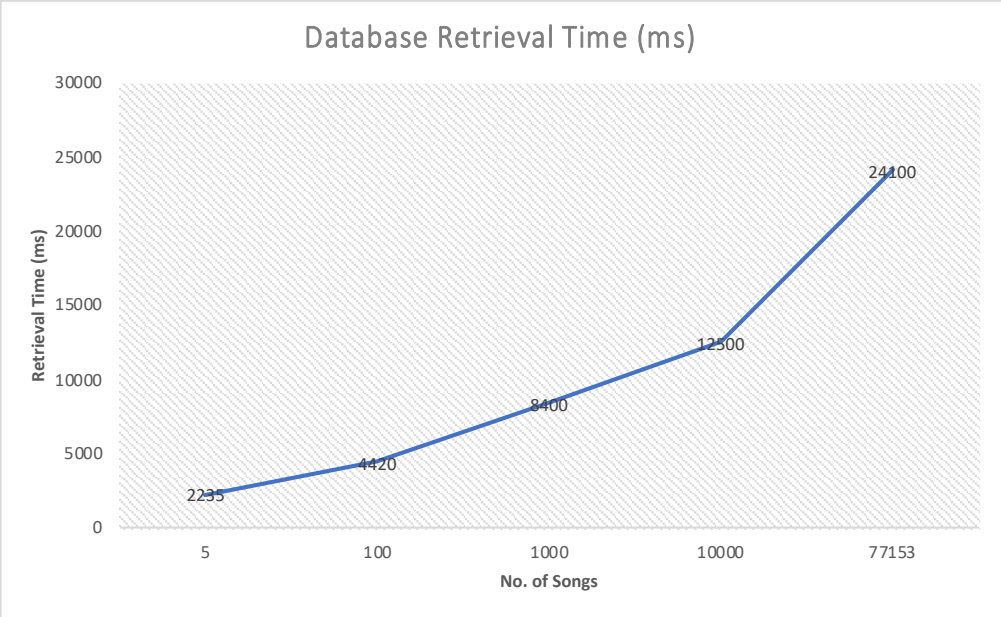

Figure 11: Database Retrieval Time (ms) against a different number of songs.

6. CONCLUSION

A review of this paper in comparison to related works suggests that QBH shows to be a viable approach for the field of MIR. A summary of the experiments conducted during this project is provided below.

The first experiment consisted of using MFCC which was not a suitable approach because, for a sample of 5 songs, it had an MRR of 0.54 compared to the MRR of the proposed approach which was measured to be 0.82. Moreover, the database retrieval time for the MFCC approach was drastically slow (approx. 1 minute). As a result, it was concluded that MFCC is not suitable for QBH systems due to the increase in retrieval time as the database scales.

Previous approaches such as Audio Fingerprinting have a better retrieval time (between 5–500 milliseconds) for 20,000 songs in comparison to the proposed QBH approach which had a retrieval time of between 3–6 seconds. However, the Audio Fingerprinting method is not a viable solution for QBH; hence, it was considered unsuitable. DL methods that were used by J.-Q. Sun and S.-P. Lee [13] produced an MRR of 0.82 for 200 songs; however, the proposed system has already passed this potential with an MRR of 0.90. However, the research in the DL system used different songs with a different sample of people for their experiments, whereas the proposed system achieved a high MRR subjected to a particular sample of people and dataset. The two main problems with a DL system are overfitting to the trained voices of people, and the gathering of the dataset to be trained and tested upon. It is a huge workload to create or find a large training dataset for a DL system for QBH. This is a great approach; however, it is only suitable if

there is a large training dataset available, and if background noise has been factored into account when training the neural network.

From our experiments of gender-based queries and considering professional/non-professional singers, we conclude that, for 100 songs, our average male sample had an MRR of 0.84, compared to the average female MRR of 0.80. Furthermore, professional singers had an MRR of 0.90 for 100 songs whilst non-professional singers had 0.72. This is expected because professional singers would hum the song in a rhythm that more closely matches the song. Therefore, the professionalism of the singers is a major factor that significantly affects the MRR of the proposed QBH approach.

Another experiment that was conducted was the measurement of database retrieval speed. For the proposed QBH system, the retrieval time scaled exponentially by O(nlog(n)) according to the number of songs in the database. For example, 77,153 songs took 24.1 seconds which is not reasonable in a practical environment.

The QBH system can have different algorithms; however, the most effective would be DTW due to its fast-paced data matching. In contrast, feature vector is most suitable for a large-scale database as DTW is a dynamic programming (DP) method and is computationally expensive. As a result, a sacrifice on the trade-off has to be made between large-scaled database capability or retrieval efficiency. Although, the MRR values were high for the proposed system, the retrieval speed can be potentially improved.

Finally, the MIR field needs to be explored more to find efficient techniques, as QBH is a current novel approach and it lacks in the data pre-processing areas. This is because the whole area comes under NLP (Natural Language Processing) which has noise

in its input that needs to be removed before passing it as a query. Background noise is only one factor, as several other challenges need to be tackled. Moreover, music retrieval can be combined with music recognition by using ML to train a model using the humming data and recognize and retrieve the song related to it.

7. FUTURE WORK

We propose a future work of using this system with integration to an ML or DL model that can be trained on previous humming data to identify new/unknown hummed queries and the song in correlation to it. The model can be further optimized by adding hidden layers and changing parameter values to match the humming audio and by introducing a larger dataset to achieve better accuracy.

Additionally, before the humming audio has been parsed, it could go through an ANR module to reduce the noise in live input. The output of the ANR module would be a clean humming sound.

Data pre-processing could be used before extracting the features from the MIDI file to remove the unwanted part of the original audio and the hummed query. Therefore, the revised system architecture would contain an ANR and data pre-processing module that would ensure that no unnecessary sound is being used for checking similarity; hence, increasing the accuracy of the similarity search and retrieval.

Furthermore, the proposed system can be improved by creating an efficient strategy such as optimizing the indexing for the database. Indexes are a great way to organize and locate data easily. It increases the speed of the data retrieval; hence, making it more efficient. Indexes reduce the number of data that have to be scanned to find the correct song

record. Therefore, to create a better QBH system, indexing can be used for the database in order to retrieve the song more efficiently.

LIST OF REFERENCES

[1] R. A. Putri and D. P. Lestari, "Music information retrieval using Query-by-humming based on the dynamic time warping," *2015 ICEEI*, Denpasar, 2015, pp. 65-70. doi: 10.1109/ICEEI.2015.7352471

[2] K. Adamska and P. Pełzyński, "Melody recognition system," *2012 Joint Conf. NTAV/SPA*, Lodz, 2012, pp. 115-118.

[3] T. Jie, L. Gang and G. Jun, "Improved Algorithms of Music Information Retrieval Based on Audio Fingerprint," *2009 Third ICIITAW*, Nanchang, 2009, pp. 367-371. doi: 10.1109/IITAW.2009.110

[4] M. Antonelli, A. Rizzi and G. del Vescovo, "A Query by Humming System for Music Information Retrieval," *2010 10th ICISDA*, Cairo, 2010, pp. 586-591. doi: 10.1109/ISDA.2010.5687200

[5] N. Kosugi et al. "Music retrieval by humming-using similarity retrieval over high dimensional feature vector space," *1999 IEEE PACRIM. Conf. Proc. (Cat. No.99CH36368)*, Victoria, BC, Canada, 1999, pp. 404-407. doi: 10.1109/PACRIM.1999.79956

[6] Wei, "An improved feature extraction algorithm of humming music," *Proc. 2011 Int. Conf. on Trans., Mech., and Elect. Eng. (TMEE)*, Changchun, 2011, pp. 2500-2503. doi: 10.1109/TMEE.2011.6199729

[7] Kotsifakos et al. (2012), "A survey of query-by-humming similarity methods", *Conf. on Perv. Tech. Related to Assis. Env.* (PETRA)

[8] A. N. Silla Jr., A. L. Koerich and C. A. A. Kaestner, "A Machine learning approach to automatic music genre classification," *J. of the Brazilian Computer Society*, vol.14, no.3, pp.7-18, 2008.

[9] Y. Zhu and D. Shasha, "Warping indexes with envelope transforms for query by humming," in *Proc. of the 2003 ACM SIGMOD Int. Conf. on Mgmt. of Data*, New York, 2003.

[10] A. Wang, "An industrial strength audio search algorithm," *Proc. 4th Int. Conf. Music Inf. Retrieval*, pp. 7-13, 2003-Oct.

[11] A. Ghias, J. Logan, and D. Chamberlin. "Query By Humming," in *Proc. ACM Multimedia 95*, pp. 231-236, November 1995.

[12] N. Mostafa et al. “A machine learning based music retrieval and recommendation system,” *Proc. 2016 Tenth Int. Conf. on Lang. Res. Eval. (LREC)*, pp. 1970-1977, May. 2016.

[13] J.-Q. Sun, S.-P. Lee, “Query by singing/humming system based on deep learning,” *Int. J. Appl. Eng. Res. 12(13)*, pp. 973–4562 (2017).

[14] "MIDI Dataset", *Composing.ai*, 2019. [Online]. Available: https://composing.ai/dataset. [Accessed: 27- Oct- 2019].

[15] "dtw", *Musicinformationretrieval.com*, 2019. [Online]. Available: https://musicinformationretrieval.com/dtw.html. [Accessed: 27- Oct- 2019].

[16] "Best audio file format for your next music recordings", *Magroove Blog*, 2019. [Online]. Available: https://magroove.com/blog/en-us/best-audio-file-format/. [Accessed: 27- Oct- 2019].

[17] *DPA*, 2019. [Online]. Available: https://www.dpamicrophones.com/mic-university/facts-about-speech-intelligibility. [Accessed: 27- Oct- 2019].