Music Search Engine Using Audio Input

Nimila Amarasinghe, Nuwan Senaratne, Dinesh Senaratne, Greshan Kinsly and Shehan Perera
Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka.

Abstract— **Today we are living in a world where music has become a trend. The various preferences over songs have made the music world more complicated. This complexity has made the searching of songs more difficult. People have to remember metadata of songs in the case of searching a song using most of the current song search engines [1], [2]. But that is not an easy thing since there are billions of songs in different languages and people naturally cannot remember the lyrics of songs. People naturally used to remember the melody of songs rather than its words. So, there is a necessity of having a song searching method with the melody of songs. This research paper discusses the steps of a process of searching songs using melody. People can give the melody of a song as the input to this process and search for songs with similar melodies.**

Keywords— **Frequency Pattern Matching, Musical Notes Approximation, Music Search Engine, Melody Transcription, Acoustic fingerprinting**

I. Introduction

Music is a universal language. Hence it has evolved simultaneously with the evolution of the human being. Due to the rapid developments that have taken place and are still taking place, music has become an industry in the modern world. The World Wide Web came up with different music search engines [1] to fulfill this vast demand to a certain extent. Most of those existing music search engines require metadata (Title, Name of the artist, lyrics, Track details etc.) of the songs as the searching queries [1], [2].

Naturally the human can remember the melody or the rhythm of a song rather than the Meta data of it. Most probably the users of those music search engines can't remember the metadata related to the songs that they want to search. Also there are people who prefer music and songs of other cultural and regional languages which they do not understand. Also some song listeners hear only a part of a song that they desire and they may not be able to find the song from the existing systems with those limited details they have. For such scenarios the music listeners face a lot of inconveniences. Hence the music listeners desire to have a proper music search engine which can take an audio input of singing or humming [2].

Properties of music have been considered to generate this method and more priority is given to the properties which make a great effect with a song [2]. Frequency is such major property which can be used to identify a song when we hear.

Our system is addressing this area of music searching which provides compared song list for a given voice input. The overview of searching songs in our system is shown in the Fig 1.

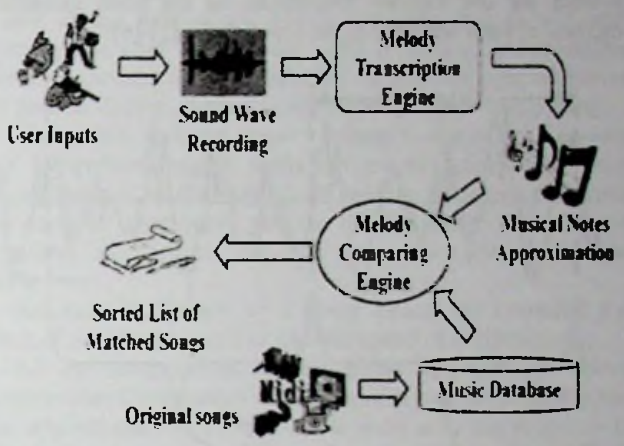

Fig. 1. System overview

II. Melody Transcription

When addressing the problem of preparing a music search engine using audio input, consideration must be given to the unique feature extraction [2], [3] from the music that reside in the database. The concepts of acoustic fingerprinting [2]-[5] can be use in that kind of scenarios.

In simple terms acoustic fingerprinting is extracting unique features from audio files which can be used to distinguish a particular audio [2], [4]. Those unique features can be extracted while preserving the major parameters of acoustic fingerprints such as Robustness, Reliability, Granularity, Size of the fingerprint and the Search time [4].

Out of the three main properties of the sound (Amplitude, Timbre Quality and Pitch) the Pitch (Frequency) is the unique feature to distinguish different audios. Hence our system performs some frequency analysis process in order to extract the fundamental frequency, as the first step of music information retrieval. This feature extraction process acts in 3 steps, segmentation [3], [5], [6], frequency extraction [5], [6] and converting to musical notation [6].

A. Segmentation

Our system takes audio input as WAV [7] file which has two channels (Stereo) and 16-bit samples in the rate of 44.1 kHz [6]. In the first step the audio input is divided into segments. This division is similar to the conversion of the continuous audio signal into a digital signal which provides simplicity in frequency extraction [8]. (Obtaining the frequencies of each point in the continuous wave is infeasible.) The main factor in segmentation is the segment size [3], [6]. The segment size should be preserved significant amount of spectral characteristics of the audio signal. Hence our system uses some milliseconds (approximately 90ms) as the segment size to reduce the disturbance for the continuous audio signal while keeping a standard pitch resolution of the adjacent segments. Then audio signal with the logical segments is

subjected to the frequency tracking process to obtain common frequency value for each segment.

B. Frequency Extraction

The frequency extraction module performs a mathematical operation called transform; which shifts the input audio signals which in time domain to the frequency domain. In our system we use Fourier Transforms as the pitch extracting method of each segment in the input audio [3]-[6].

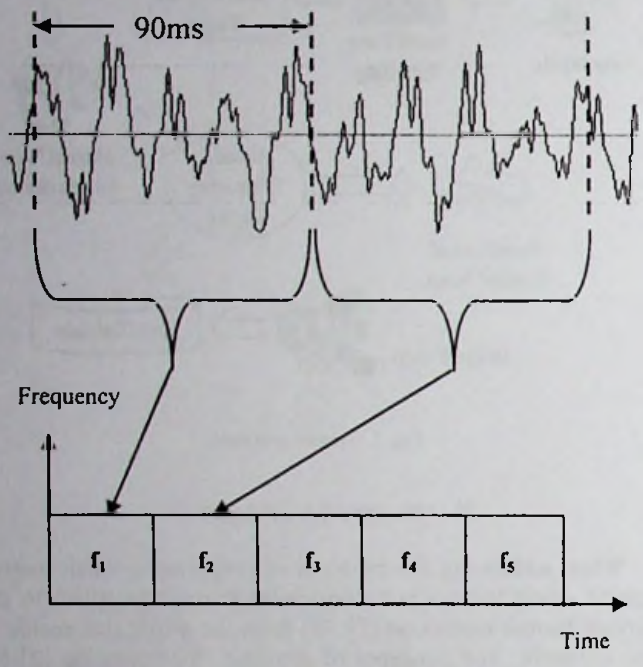

Fig. 2. Melody Transcription

As illustrated in the Fig. 2 all information in each segment passes through the pitch tracker and it obtains a common frequency for each frame [6]. Those pitch values are stored in a system specific vector which has time as the independent factor while pitch values as the dependant factor. Frequency extraction process can also be named as the melody transcription engine because we are actually distilling the melody of the audio clip as frequency vector [6], [8]. It is possible to use this result vector as a basic acoustic fingerprint due to the existence of all parameters which prefer to be occurring in a robust audio fingerprint. Due to several practical issues in general user inputs this frequency vector cannot be used directly in the comparison process. Hence this basic audio fingerprint subjects to further processing to improve its uniqueness to distinguish itself among other audios.

C. Rounding to Musical Notes

The result frequency vector of the frequency extraction process is further processed by rounding the frequency values of it to nearest musical note's frequencies. Generally the exact comparisons between the original songs and the user queries are infeasible due to existence of random errors in the user hummed clips. Hence this process is followed in order to change the random pitch values in practical frequency vectors to nearest standard values. Following real world example illustrates how the random errors in user inputs handled within this phase of acoustic fingerprint extraction process.

Assume, Time vs. Frequency graphs of two audio clips of same song sung by two different users.

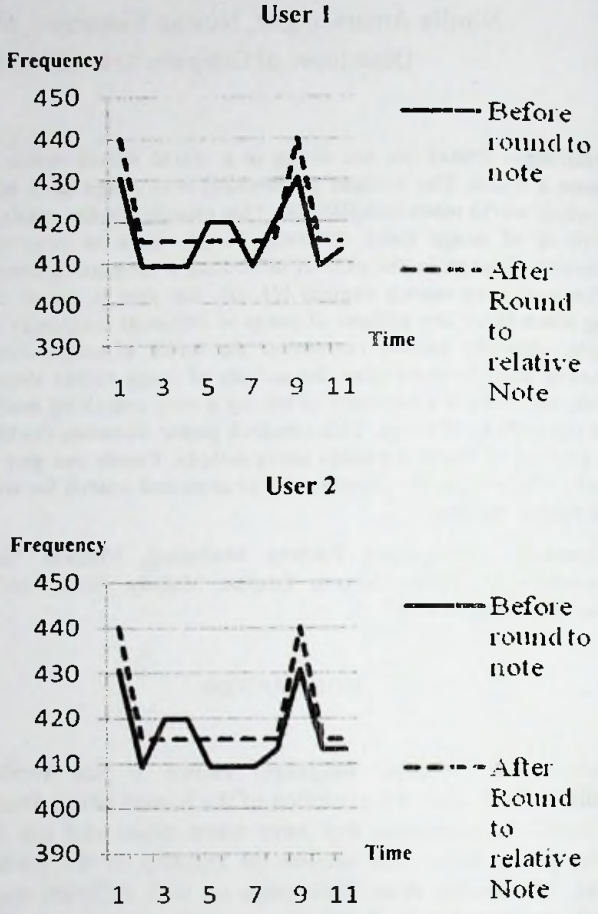

Fig. 3 Effect of rounding the frequency values to nearest musical notes

As in Fig. 3 although the patterns of the same song sung by two different users are differed from each other, the patterns that obtained by rounding to nearest musical notes is exactly the same. Likewise this process reduces the randomness of the user inputs and makes them more unique.

III. Melody Comparison

Generally when someone is going to prepare a music search engine which take audio inputs, there are three main concerns to consider by the designer of the system. Those are the pitch issue (Differences in pitch values between the user input and the relevant original song), the tempo issue (Randomness of the tempo in the different parts of the user input) and existence of musical instrumental parts in the original song. The methods of overcoming those problems are discussed within the description of the searching algorithm of this paper.

The songs database is formed by passing all the original songs through the audio fingerprint extraction process and storing only the musical notes approximated frequency vector along with the other details of the song. Also when a user provides an input it is also following the same process before getting into melody comparison engine.

A. Melody Comparison Engine

In this phase all the fingerprints of the original songs fetch to the main memory and compare with the user input one at a

time. Actually the comparison follows a method called frequency pattern matching in order to find the melodic similarity between the songs. Finding the exact place of the original song where the user starts singing is another important factor to be considered. In order to find this exact place our system use sliding window mechanism of user query along with the original song.

Another important action is performed to extract the vocal frequency range from frequency vectors of both the original songs and the user queries before focusing on the comparison between them. We use the range 90Hz – 800Hz. This extraction do not totally eliminate musical instrumental parts rather reduces the effects of it as well as the effect of background noise of both, while preserving the original time spans of them.

Suppose we have two frequency vectors, one for hummed song and other one for original song. Our objective is to compare the hummed song with each of the original songs in the database [6] and get some accuracy value according to the comparison strength. If the comparison gives higher accuracy, it means the hummed part and original song part matches well. So, according to the accuracy, we can sort the result song list such that the songs that have heights score in the top of the result list.

In the first step the initial frequency segment of the user query compares with the first segment of the original song and calculates a frequency ratio. This frequency ratio is calculated to find the frequency difference between the frequency regions of the original song and the user query (A song can be sung in different musical octaves). An expected value is computed using this frequency ratio and the frequency value of the second segment of the user query. Then the calculated expected value compares with the second segment of the original song. Most probably in real world examples (Fig 4) those expected values are different from the original value. So the algorithm calculates another parameter called pitch variance, by getting frequency ratios between the expected value and the corresponding original song value.

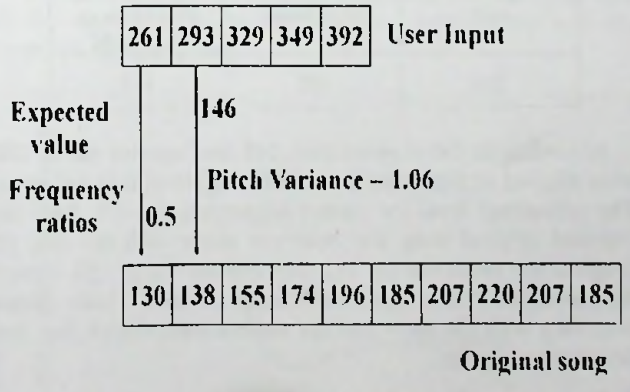

Fig. 4 Process in the melody comparison engine

The pitch variance value 1.06 in Fig 4 is a constant for frequency ratio of adjacent musical notes. We use this property to overcome the pitch differences between user queries and the original songs. If the pitch variance value of the comparison lies within the defined threshold boundaries a score value is adding to the score of the current iteration, by the score function with the consideration of comparison pitch variance value. Also this comparison is considered as a matching point.

This process continues until either user input frequency vector ends or non- comparison point occurs. The process moves to the next iteration which starts comparison from the second segment of the user query, if the previous activity ended by ending the user query vector. This new iteration system allows sliding through not only the original song but also the user query as well which helps to overcome the initial error of the user query (Initial parts that have low level of voice of the user query which occurring due to not start singing just after the recorder starts recording). After ending the slide through the user input the comparison process moves to general sliding window which slides on the original song.

If the pitch variance doesn't belong to the defined regions the comparison engine shifts the normal comparison to a special king of mechanism which handles the tempo deference between the user query and the original song. Tempo issue handling mechanism keeps an assumption on the tempo differences.

Assumption: Tempo of a query cannot be exceeded the limit of double the speed or half the speed of original song.

All the comparisons in this phase consider the above assumption. If the tempo of the hummed sample is lesser than the original one, the comparison is done as in the section 1 in Fig 5. The comparison follows the other way as section 2, if the tempo of the user query is more than the original one.

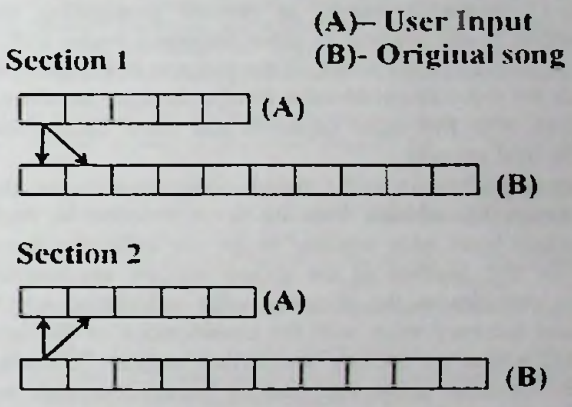

Fig. 5 Tempo issue handling process I

The same method which use in the direct comparison, is also using in this step too (Only the comparing segments are different). For example suppose the second comparison of the first iteration found a tempo issue and also it found a comparison point when following the method as in section 1 in Fig 5. The tempo issue is considered for that comparison only. In the next comparison of the same iteration the algorithm follows the normal direct comparison rather than the mechanism in the section 2. So it's just skip the first segment of the original song as illustrated in the Fig 6.

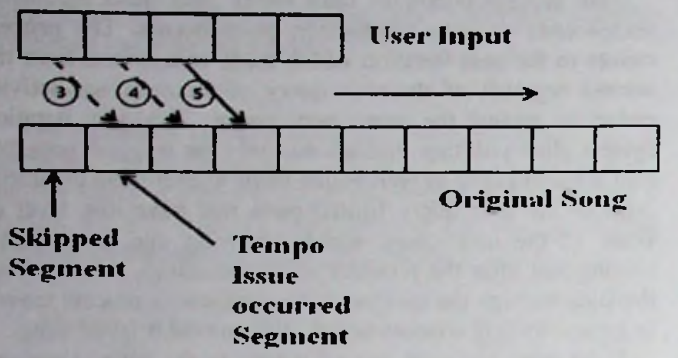

Fig. 6 Tempo issue handling process II

Fig 6 represents how the 3rd, 4th and the 5th comparison of the first iteration occur. Those comparisons done as same as the direct comparison happens and the tempo issue handling processes calls only if this progression found another non-comparison segment in either user query or the original song.

Suppose the 4th comparison of the Fig 6 found another tempo issue and also found a comparing point while following the procedure in section 1 of Fig 5. In this scenario the 1st segment of the user query is compares with both the 2nd and 3rd segment of the original song. This situation is considering as occurring of mismatch point. Our comparison algorithm keeps a threshold value for number of consecutive mismatch points (5 mismatch points) to prevent propagating the comparison of one segment in either frequency vector with a lot of segments in other vector. If the progress of the iteration exceeds the above threshold value it skips the segment of been compared with five other segments and starts comparison from the next segment.

After single iteration in the melody comparison engine, the score values that obtains from the score function for each comparison point adds together to get the collective score value for that position of the sliding window mechanism. Then it provides to the accuracy value calculation which computes accuracy value with the consideration of the data collected within the melody comparison engine. Following sections describe about the tracking parameters and the accuracy function.

B. Tracking parameters

- Number of compared points (X) (Total number of comparison for a particular iteration)
- Consecutive mismatch points
- Pitch Ratios
- Total number of matching points in the user query (Y)
- Collective score (S) in each iteration.

C. Accuracy Function

- Accuracy Level = (S/X)* (Y/Length of the user query)

Finally a maximum value is selected among all the iterations accuracy values of each particular user query-original song pair. The result song list is prompting to the users in the descending order of this accuracy value.

IV. Results

Initially most of the testing was focused on calibrating the search algorithm parameters. After the calibration some case studies were conducted to test the accuracy of the results and the performances (search time) of the system.

A. Accuracy Test

For this test we used a database which consists of 200 original songs from different artists, different music types (Classical, Pop etc.) and different languages. Also as the user inputs, we collected 200 different user queries of sung or hummed from 8 different users.

Some factors were observed during the testing and tabulated them in a standard way. The factors that were considered can be listed as follows.

- Name of the user
- User query title
- Indication whether the user query is aligned or not to the exact place.
- Obtained score value
- Place of required song in the search result
- Score of the first song of the search results(if required song is not the first song)
- Score of the 2nd song(if required song is in the first place)
- Matching duration of the user query in milliseconds.
- Matching duration of user query in milliseconds for the first song (if required song is not at the first place)

This test was mainly focused on obtaining an average percentage values for correct alignments (If the user query is aligned to the correct place of the required original song, it is considering as a correct alignment) as well as the number of times the required song came to the top 10 of the output song list. Our observations are tabulated in the following graph.

TABLE I
Results of the Accuracy Test

Number of songs in the test	Number of correct alignments	Number of required songs that came to the top 10
200	148	112

According to the observations, 148 user queries out of 200 were aligned to the exact place of the required original song. The percentage level for correct alignments is 74%. Also the required original song was retrieved along with the first 10 songs of the result list for 112 user queries out of 200. Hence the calculated percentage for this result is 56%. Those values may vary with the users and the environment which they are recording songs etc.

B. Performance Test

The performance test was also conducted with the same database that used in the accuracy test. In here we have prepared several user queries of same song which have different lengths. The results obtained from this test are tabulated in the table 1.

TABLE II
RESULTS OF THE PERFORMANCE TEST

Length of the user query (Seconds)	Place of the required song in the result song list	Time taking to provide results (Seconds)
4	32	20.60
8	2	30.14
12	3	39.44
16	4	51.02
20	4	63.83

The searching time for a given audio input is increased with the length of the input. That was one of the major observations of this test. Also the place of the required song decreased at first and then it increased with the length of the input. The following graphs illustrate how those factors getting changed with the length of the user input.

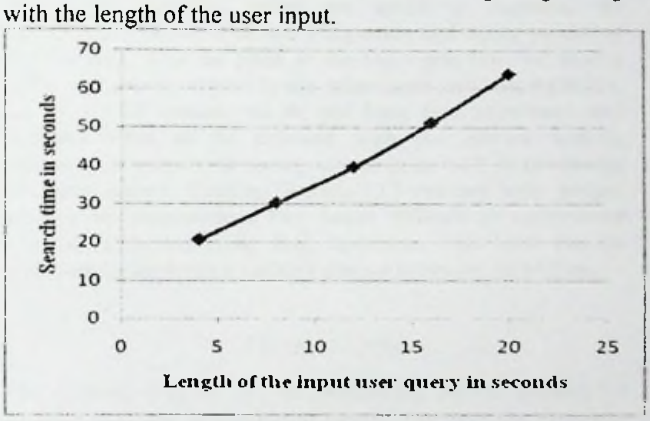

Fig. 7 Search time graph

By above graph we can understand that the searching time of the system has some sort of linear relationship with the length of the user query.

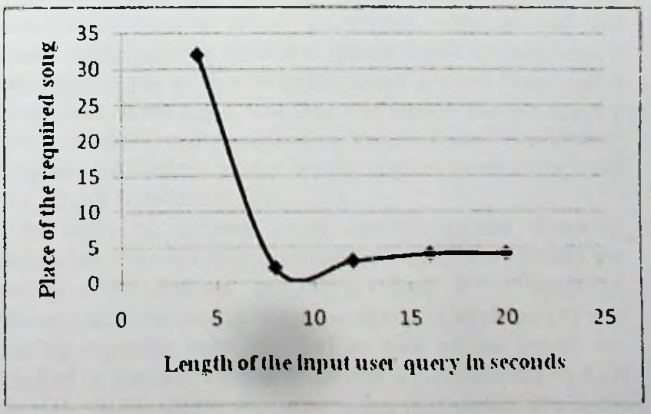

Fig. 8 Graph of the places of required song

This graph has a minimum. So as shown in the above graph we can see the maximum accuracy has come for user queries around 9-10 seconds. Hence the user has to provide the user queries which have the length about 10 seconds to get the maximum accuracy of the comparison algorithm.

V. CONCLUSION

In this approach we use the method called frequency patter--n matching using musical notes approximation. Actually this is a melody comparison method which can be used to compare two audio clips and calculate an accuracy value for the melodic similarity between them. Hence this approach can be used in developing song searching method which takes audio files as the input queries. Also this method can be used in any other scenario where the comparison of melody of the music involving.

REFERENCES

[1] Alexandros Nanopoulos, Dimitrios Rafailidis, Maria M. Ruxanda, and Yannis Manolopoulos. "Music Search Engines: Specifications and challenges," *Information Processing and Management*, vol. 45, pp. 392-396, 2009.

[2] Rainer Typke, "Music Retrieval based on Melodic Similarity," Ph.D. dissertation, Utrecht University, Utrecht, Netherlands, 2007.

[3] Arunan Ramalingam and Sridhar (Sri) Krishnan. "Gaussian Mixture Modeling of Short-Time Fourier Transform Features for Audio Fingerprinting," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 4, pp. 457-463, December 2006.

[4] Jaap Haitsma and Ton Kalker. "A Highly Robust Audio Fingerprinting System," *Journal of New Music Research*, vol. 32, pp. 211-221, June 2003.

[5] Yu Liu, Hwan Sik Yun, and Nam Soo Kim. "Audio Fingerprinting Based on Multiple Hashing in DCT Domain," *IEEE Signal Processing Letters*, vol. 16, no. 6, pp. 525-528, June 2009.

[6] Lloyd A. Smith, Rodger J. McNab, and Ian H. Witten, "Music Information Retrieval Using Audio Input," [Online]. Available: http://eprints.kfupm.edu.sa /52629/1/52629.pdf. [Accessed: October 10, 2009]

[7] P.Kabal, "Audio File Format Specification," June 19, 2006.[Online]. Available: http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/WAVE.html [Accessed: October 18, 2009]

[8] Bryan Pardo, Colin Meek, and William Birmingham, "Comparing Aural Music Information Retrieval Systems," [Online]. Available: http://www.music-ir.org/evaluation/ wp1/wp1_pardo.pdf. [Accessed: October 25, 2009]