A Highly Robust Audio Monitoring System for Radio Broadcasting

E.D. Nishan W. Senevirathna
University of Colombo School of Computing Sri Lanka
nishan.senevirathna@gmail.com

K.L. Jayaratne
University of Colombo School of Computing
Sri Lanka
klj@ucsc.cmb.ac.lk

***Abstract*—Proposing a novel approach for monitoring songs for the radio broadcasting channels is very important for the interest of singers, writers and musicians in the musical industry. Singers, writers and musicians have a claim to intellectual property rights for their songs broadcast over all the radio channels. According to this intellectual property rights act singers, writers and musicians should be paid for their songs broadcast over all the radio channels. Therefore we propose a real time audio monitoring approach to solve this problem which includes our own audio recognition algorithm. It is easy to recognize a song, when you provide the original high quality blueprint of the song as input. But we can't expect such kind of audio input from radio channels since lots of transformations are possible before reaching the end user or listener. For example, adding environmental effects such as noise, adding commercials on the song as watermarks, playing more than one song as a chain without adding any silence between them, playing a part of the song, playing same song in various speeds and so on. These transformations cause change in the uniqueness of particular song and make the problem even more difficult. The algorithm we proposing is resistant to noise and distortion as well as it is capable of recognizing short segment of song when broadcasting over the radio channels. At the end of the processing our system generates a descriptive report including title of the song, singer of the song, writer of the song, composer of the song, number of times it was played and when it was played for all songs for a particular period for all radio broadcasting channels. We evaluate our system against various types of real time scenarios and achieved overall higher level of accuracy (96%) at the end.**

Keywords-Audio fingerprint; features extraction; playlist generation; wavelets; broadcast monitoring

I. INTRODUCTION

Since the beginning of the human life sounds, voices and noises are attached to their lives. A song is some sort of arrangement of sounds in time. Usually humans like to collect songs as repositories which they call albums. The need of songs classification and identification has risen as a result of the speedy growth of these repositories. Audio recognition isn't very easy task. Therefore this is still an open research area. The song recognition problem becomes a more difficult problem when we step in to the radio broadcast monitoring. We can't expect the high quality original blueprint from broadcasted radio streams since lot of transformations and alterations are possible.

Radio broadcast monitoring is still a manual process and it requires lots of man power as well as it is still a very costly operation [1]. Nowadays, there are so many radio channels as well as huge number of songs are broadcasting on each channel in a day. Therefore the process of manual song monitoring cannot be carried out further with high quality. The importance of the novel monitoring process is that it should guarantee the copyright enforcement of radio broadcast songs. This is the problem we will take into account throughout this research.

Identifying an object by examining the meta-data associated with it, is very straightforward if that object is a text file. But we can't recognize multimedia objects easily without using proper content-based identification techniques. Content of a song is its acoustic qualities of audio such as wave pattern, beat frequencies distribution and so on. Before applying content-based identification (CBID) techniques for radio broadcast monitoring, we have to consider the following challenges.

- The radio channel may modify the original song by adding commercial or some other unwanted things.
- A part of a song may be missed or chain of songs may be played continuously.
- Audio signal can suffer from several transformations (such as noisy) before reaching the listener.

Our solution considers all of these challenges and we propose a highly robust novel approach in order to solve this problem.

Audio fingerprinting is a well known approach, which uses to identify a particular audio clip [1][2]. Audio fingerprinting is a CBID technique.

Two approaches of audio fingerprinting differ from one another from the feature extraction. We propose highly suitable feature extraction technique for this audio broadcast monitoring problem.

Finally we prepare a summary report as the output which contains all necessary details such as available metadata such as author of the song, musician and singer, time of the broadcast, information about the channel and so on.

Rest of this paper is organized as follows. Section II provides a review of the related background. We discuss the design of the system in the section III. We prove the achievability of our proposed solution under the implementation in section IV. Then we will evaluate the proposed system using obtained results in the section V.

DOI: 10.5176/2251-3043_3.2.250

Finally we mention our concluding remarks and suggestions for future works in section VI and VII respectively.

II. Related Work

According to the literature, there are number of applications for audio fingerprinting [1] like identify a song using mo- bile phones, filtering technology for file sharing, automatic music library organization, automatic play-list generation and broadcast monitoring and so on.

A playlist may define as a finite sequence of songs which is played as a complete set [3]. Playlist can be seen as two point of view. Firstly songs listener prepare playlists before songs are actually played. Secondly songs broadcasted over the radio channel is interest to prepare playlists after songs are actually played. In this research we consider the second one. Next, let’s consider well-know audio identification methodologies in this research area.

A. Radio Data System (RDS)

Radio Data System (RDS) is a communication protocol which uses to transmit information such as song’s metadata. RDS enable receivers can view embedded RDS messages. The European Broadcasting Union (EBU) member countries introduced this technology and mainly used to transfer road traffic information to FM car radio [4][5]. We can use same concept for broadcast monitoring.

Required meta-data are encoded to FM carrier wave during the modulation stage. Then receiver can decode the message and access these meta-data. We can use this approach to generate broadcasted play-list but RDS is not enabled in most of countries. Further, RDS system has some drawbacks [6].

The radio data signals must be suitable to transfer RDS messages. It means they must not make any interference to the reception of sound. To access RDS data reliably, recipient should have better signal. Usually particular message is embedded to one or more places on a song, therefore part of a song may not be sufficient to identify the song[6]. However we can use RDS technology to generate playlists.

B. Audio Watermarking

Audio watermarking technique is almost similar to RDS system, it works embedding particular message without altering the perception of the sound. We can use this to detect copyright infringement and variety of other applications such as broadcast monitoring, owner identification, proof of ownership and so on [7]. Broadcast monitoring is one of the major applications of audio watermarking. Any interested third party can use audio watermark to identify particular song or any other audio clip. Audio watermark is put into particular song prior to broadcast. Usually watermark cannot be removed or changed by a third party.

However, watermarking approach consists of several drawbacks which restrict the usage [8]. Really inaudible watermark development is very challenging task especially when we are dealing with compressed audio formats like MP3. As well as, there is no choice for already released material. According to the section A and B we can say that RDS and audio watermarking approaches consider specific features (such as a message) or additional information which is embedded to an audio clip. Because of that, we can’t use these methods for any audio song.

C. Content Based Audio Identification

Content based identification techniques are based on the acoustic qualities of audio. Different feature extraction mechanism needs different system implementations.

Audio fingerprinting is a well known but emerging content based audio identification technology. Our solution is also based on the audio fingerprinting technology. First acoustic relevant characteristics are extracted from a given audio content and then they are stored in the database in compressed manner. Figure 1 shows the flow of the Audio fingerprinting approach. Audio fingerprint is a compact representation or signature of full quality large audio object. It is exactly similar to the crypto graphical hash value of a large message. Instead of compare large audio object we can compare small fingerprint in order to find the most similar audio object [1]. According to the [1] there are several advantages of using audio fingerprint instead of large audio file itself.

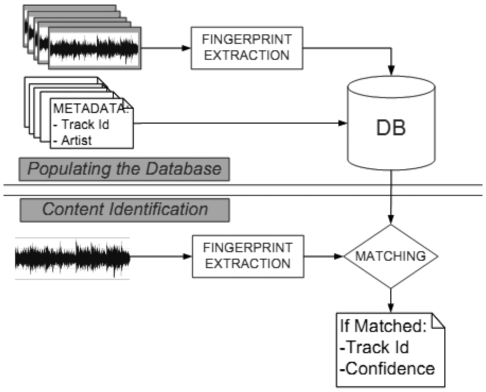


Figure 1. Flow of the audio fingerprinting approach. This consists with major two stages, populating the database and content identification. In the populating the database stage, we extract fingerprint from original high quality completed songs. Then they are stored in the database with the related metadata. After this stage we can identify the songs (content identification stage) which are in the database. We extract fingerprints again from the unknown segment of song. Then those are directed to matching process and recognize the highest confidence one [9].

- Audio fingerprint takes very law memory to store therefore it reduces storage requirement
- Since audio fingerprint is too small, it makes efficient comparison
- Perceptual similarities should not consider when two fingerprints are matching as perceptual irrelevancies have already been removed

According to the [1][4], we have to consider sets of properties when we are introducing new audio fingerprinting approach. These are accuracy, reliability, robustness, security, versatility, scalability, complexity and so on.

According to the most of the researches any audio fingerprinting approach consists of several major steps, and Figure 2 shows the flow of these steps.

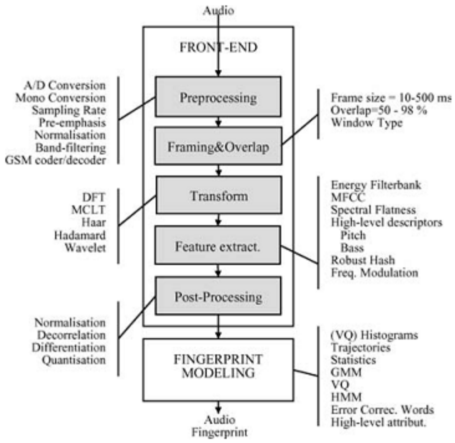


Figure 2. Major steps of the audio fingerprinting. One approach differs from the others from implementations of each step. Therefore accuracy and reliability depend on the implementation. Preprocessing stage highly depend on the application it may be completely ignore in some cases. In the framing stage, divide input signal into several overlapping frames. Then apply some sort of transformations such as short time Fourier transformation. Then extract some unique features after this step we can transform the output in order to satisfy our final objective [10].

D. Concept of AudioDNA

This is another content based audio identification methodology. This is highly scalable system for the automatic identification of audio [8].

Any audio stream can be seen as sequences of acoustic events. AudioDNA approach replaces every acoustic event by a letter in order to obtain audio gene. Such sequences use to identify a song or compare two or more audio songs together. AudioDNA is a sequence of audio genes which are extracted from a piece of music. Most of the things are much closed to audio fingerprinting schema but searching or matching algorithm is different from each others. AudioDNA matching algorithm consists with two main processing steps, exact matching and approximate matching like audio fingerprinting. Both of them use FASTA searching algorithm which is most effective practical searching method of biological gene sequences.

III. System Design

In this section we will discuss the design of our approach with solutions for already realized problems. We divide this section into set of sub parts in order to discuss major steps clearly.

First of all we have to realize clearly the problems, we are going to solve. These problems (challenges) are the design considerations. We tackle each and every challenge step by step. Followings are the design considerations in our design.

- Songs may be destroyed by noise or any other external effects
- Commercials or talks may be added while playing a song
- Part of a song may be broadcasted
- Distinguish songs from other items such as advertisement
- Scalability issue, i.e. our design should be able to handle millions of songs

In order to tackle above considerations we design our approach as the Figure 3 which shows the flow of the design.

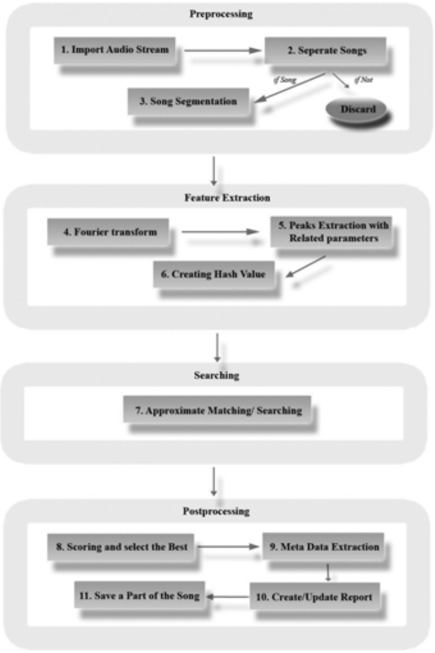


Figure 3. The overall flow of the design. There are four major parts, each one can be divided further.

A. Preprocessing

We transform the raw radio stream into some sort of audio object/s which can be manipulated easily. We use the audio sound card line-in hardware interface to import the audio stream into the working environment. Radio channel should be tuned properly, however noisy can be added to the stream. But it is not a problem since these are not hold for a longer period. Short time noisy will suppress gracefully by the proposed system. To buffer audio stream we use following strategy. We take 5 minutes long audio stream and then it is processed. While it is processing we collect simultaneously next 5 minutes and so on. To do this we use two threads, it will help us to hold performance at optimal level and hold buffering process steadily.

We separate songs from the mixer of other audio objects. This helps us to increase the system efficiency. To separate songs from the others, we suggest very simple method, even it is failed proposed overall system will work normally. We have observed that the most of audio objects consist with set of frequent silence except song object. By analyzing the pattern of these silence and the time duration between two adjacent silences we can distinguish songs from the others with high accuracy. We can't define a silence globally since it is depended on the strength of the stream. Therefore we consider the mean square root signal of the original signal then define a silence as a local value to the current audio frame.

To address the above mentioned design considerations, we frame the extracted song into 40 seconds long segments. Then each segment is processed. Following above strategy we can reduce the effect of unrelated things such as noisy and commercial. Here we assume that at least one frame has good quality. It is valid assumption since if we can't find at least one 40 seconds long good quality frame out of about four minutes long song, then it should not be a song at all.

B. Feature Extraction

As the first step, we perform Short Time Fourier Transformation (STFT) in order to convert time domain signal into frequency domain. The reason is that we can easily manipulate, extract frequency related features from a frequency domain representation than time domain.

We extract peaks and other related parameters. As we all know noisy and distortions are obvious in the radio broadcasting. Therefore we should find candidate features that could survive in the noisy or distortion environment. According to this [11] we realized that spectrogram peaks can be survived in presence of noise, but if the audio clip is faced to a distortion then peaks may not be survived. But it is not a big problem since framing process will help us again to suppress this problem.

Therefore considerable amount of frames are remain unchanged, these frames can be used to identify the song and distorted frames are automatically omitted. When inserting a songs into the database, we can take original ones since it is controlled process. But in the matching process, unknown song (or part) can come from anywhere in the original song.

Therefore matching process should function properly independent of the position. Consider the Figure 4, it shows two adjacent peaks. t1, t2, f1, f2, Δt and Δf are unique values at the most of time [11]. Using these values we create a hash value. If some amounts of hash values are matched correctly then we can conclude that the considered two objects are same.

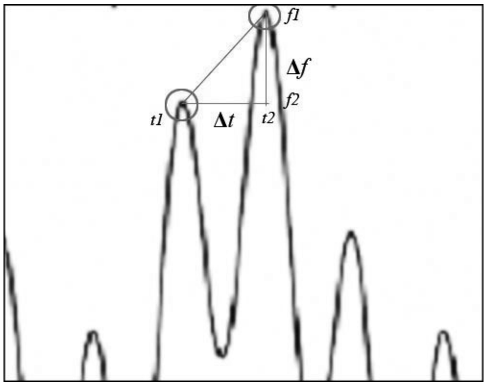


Figure 4. Two adjacent peaks and other related parameters. We use these parameters to create a unique hash value.

C. Searching

The input to this stage is a set of hash values. Our goal is to find out the most similar database track for these values. But the challenge is that these two set of hash values may not be matched 100%. Because of this nature, we use the term approximate matching. Assume that unknown song is converted into N number of hashes. In this stage, matched song ID is obtained for each and every hash from set N. After that obtained result is analyzed to identify the unknown song. We use scoring based approach to extract most matched song.

Assume that we have inserted 15,000 songs to the database. Each song is divided in to 7 frames as well as 50 hashes are extracted from each frame. According to this database, total number of hashes is 15,000×7×50 = 5,250,000. Even if brute force search is possible to find a matching hash against 5,250,000 hashes, it is not practically achievable. Since this takes considerable amount of time as well as resources. Therefore we should introduce new approach instead of brute force search. There are 5,250,000 hashes in our data base, number of unique hashes equal to 2^{20} = 1,048,576 if hash value consists with 20 bits. It is approximately one fifth of the database. The other thing is number of unique hash values independent from the number of songs. According to above observation we suggest a database structure like Figure 5.

D. Post processing

We extract metadata related to the song and then generate a complete report which contains metadata like song title, singer, musician and some other important information such as when it was played and duration of the song and so on. We extract a small segment of actually matched song when it was playing. We can use that clip to verify the matching process, whether it is correct or not. But it is a manual process.

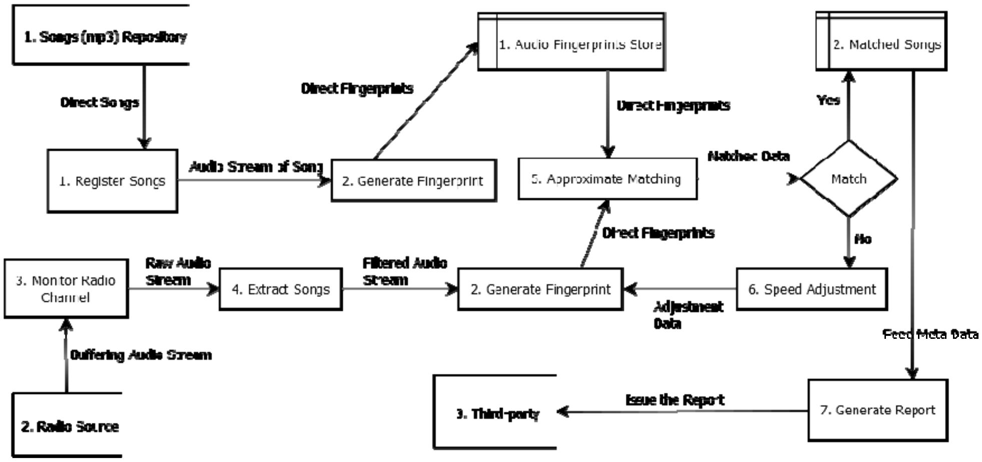


Figure 6. The system flow diagram. There are three entities song (mp3) repository, radio source and any third party. First two entities supply the inputs to the system and third entity interest to get the output. Adio fingerprint store and matched songs are data stores, others are system processes.

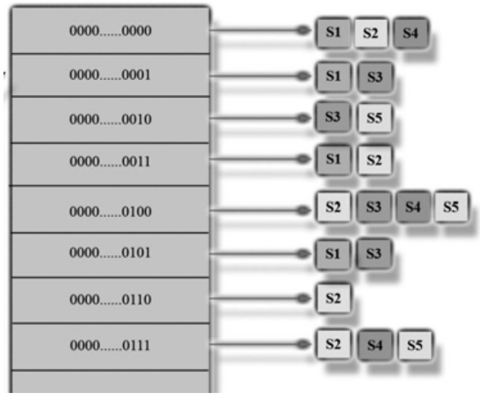


Figure 5. Proposed database Structure. This is the look-up table design for speed matching. One or more songs can attach with a same hash value. When we are matching a hash value, we can get sets of songs related to it. Continuing this process for all hashes, we can score sets of songs and finally can get the most valued song.

IV. Implementation

In this section we will discuss the implementation details in respect to the proposed approach in design section. Flow of the implementation is almost similar to the flow of the design. Consider the Figure 6, it shows the flow of our system. In this section we will discuss only major process since most of process were discussed during the design section.

There are two major parts, the first part is the song registration and the second part is the matching part. Consider top left corner of the Figure 6. Songs (mp3) repository, register songs, generate fingerprint and audio fingerprint store are belonged to the first part and rest of objects belong to the second part.

A. System Entities

There are three entities of the flow diagram, refer the Figure 6. We will discuss each and every entity.

1) Song (mp3) Repository: before going to the matching or monitoring process we have to register songs. This is the main input entity to the system which consists with huge number of songs in mp3 format. Mp3 format is the mostly available song format, therefore we use this format as the input song format.

2) Radio Source: this is also a input entity to the system. We can use any radio channel's output as input audio stream to the system. As we already discuss, we use sound card line-in interface to connect a radio output to the system. Instead of it, we can use external or USB radio card adapter (or TV card adapter which is consisted with radio). The second option is more user friendly and can be used easily. If this radio source can give high quality output then we can expect highly accurate results from the system. However we can't prevent the external effects such as noise, but fortunately these kinds of effects can be handled by the proposed system.

3) Third Party: this is a report which can be use as the system output. Any third party who is interest about this report, such as singer, writer of the song, musician and so on can use this report for their reference.

B. System Data Stores

There are two data stores. Major data store is the fingerprint store. We prepare sets of fingerprints at the end of song registration process and store them for future usage. Figure 7 shows the architecture of this data store. After obtaining correct match, we will store these information in a data store which is the second data store in the system and has been denoted as ”Matched songs” in the Figure 6.

C. System Process

1) Register Songs: Under this process, we insert (register) songs to the database. We can match only registered songs from the system. Therefore we can use pure, high quality and completed versions of songs.

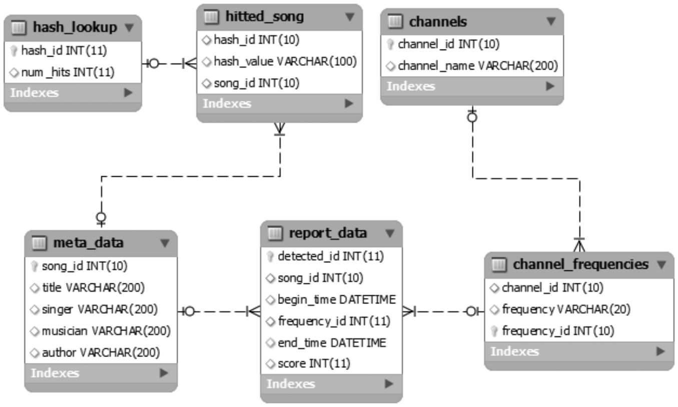


Figure 7. Database model

During this process, we collect related metadata, which will be shown in the final report. Each song is assigned a unique id which will be used by other process.

2) Generate Fingerprint: This is the most important process of our system. We developed our approach based on the idea we proposed in design phase [11]. As we mentioned in design phase, peaks are the most stable feature in any kind of audio object. To extract peaks related features, we follow set of steps.

First we perform STFT on each frame. Figure 8 shows all the parameters we used for this. Then we calculated frequencies on each window and created a complex matrix which contains set of frequencies against the time. In other word this matrix contains 256 (overlapping samples) rows of frequencies and K columns where,

$$K = \frac{\text{lengthofsignaloverlappinglength}}{\text{length(window)}-\text{overlappinglength}}$$

Before move to the next step, normal frequency values are converted into log scale ones to mitigate the large scale variations. Then, we take the local peaks frequencies and create a vector which contains all local peaks. To extract global peaks we used following method. If an element is greater than both top and bottom element then it will be taken as a global peak. In Figure 9, it shows a particular column (for convenience it is represented in horizontally).

Then we combine extracted peaks frequency and local time differences in some manner in order to create unique hash value. Thereafter we store these peaks in a database table.

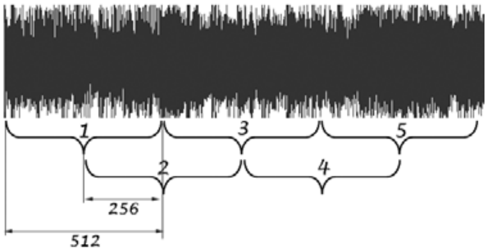


Figure 8. Parameters for STFT. We used 512 length windows with 256 overlapping samples. Then frequencies are calculated on each frame.

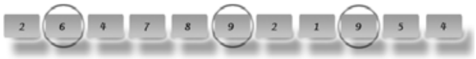


Figure 9. Local peaks extraction. If an element is greater than both left and right element then it is taken as a local peak. Elected local peaks are shown as a circle.

3) Monitor Radio Channel: This process is very important since overall system accuracy is depended on this. Once we started to monitor or buffer particular radio channel, then it should be continue until the radio is switch off. However at the meantime, short time buffered audio clip has to be processed. Otherwise at the end of some period of time, (for an example at the end of the day) we have to deal with very large audio stream (i.e. 24 hours lengthy audio clip). Because of this issue, we take five minutes long audio clip and then it is processed before considering the next buffered audio clip. But this approach will introduce another problem. If we follow this approach then we can't obtain continues audio stream since to process a five minutes long audio clip it will take some period of time, for an example one minute. According to this, it will lead to miss one minute long audio clip before we buffer next 5 minutes long audio clip.

As a solution to this problem, we used two simultaneous threads, one thread is responsible for buffering five minutes long audio stream and other one is responsible for processing buffered clip simultaneously, refer the Figure 10.

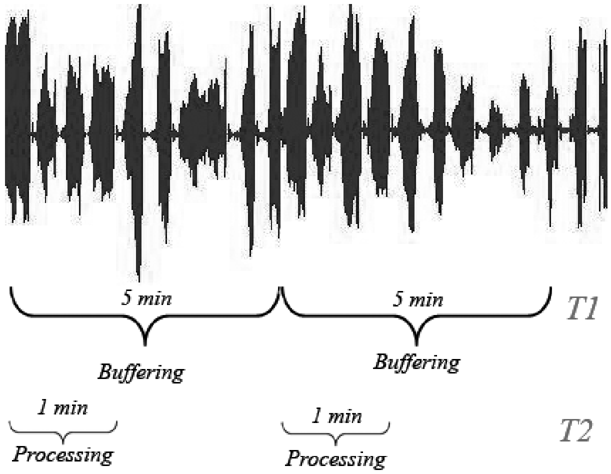


Figure 10. Buffering and Processing Using Two Threads

4) Extract Songs: The input to this process is a 40 minutes long audio clip which may be a part of a song,

a part of a drama, a part of a conversation, a part of a talk or a commercial and so on. If we direct each and every frame for further processing, then it will consume system resources unnecessarily. Our main interest is to identify the song. Therefore, in this module, we always try to filter out only the frames of the songs from the other audio objects and direct only those frames for further processing. We used no. of silences based approach to extract songs.

We used very simple technique to do this since we need not expect 100% accuracy from this module. There is another barrier which is song detection process. If a non-song object is filtered out then it will not be matched with any song in the database. Therefore we expect from this module to increase the system efficiency. As we discussed in the design phase, we consider the number of silences which are laid on 40 seconds long frame to distinguish the songs. What is meant by silence?

Silence: audio signal should be spread below to some level (it may not be a fixed value) as well as it should be hold some duration of time.

What we have done here is that calculated the average sample value for 40 second long clip. And we consider positions which are laid below to this average level (0.25 seconds) as silences, refer the Figure 11. If there are more silences than predefined threshold then it is omitted as non-song objects.

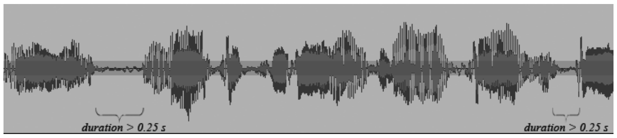


Figure 11. Detecting silences. Area of the transparent bar denotes the averaged signal and we can see there are about two silences in this clip according to our silence definition.

5) Speed Adjustment: In this section we discuss the approximate matching and selecting process. At the 99% of time we can’t expect perfect matching between unknown clip and original clip which is in the database. We discussed the reasons for this in the introduction. Consider an unknown audio clip and the most matching clip for that unknown clip. We can coincide these two audio clips to observe the similarities. Figure 12 shows such a situation. Positions like P1 and P3, two clips coincide properly (no considerable dissimilarities). But two clips are not coincided during the position P2. Practically we get these kind of situation most of the time (actually we get the mixer of P1, P2 and P3).

What we did here is that consider only matching peaks and present it as a percentage against the number of overall peaks. Through vast number of experiments, we obtained some threshold value. If matching percentage is greater than or equal to this threshold value. we extract it as a matching clip to the given unknown clip. There may be more than one matching clips, if so we extract the highest score one. Suppose there is no at least one matching clip then what we can do. This situation is possible due to number of reasons; we will discuss this problem in the next section. If there is no at least one matching clip then we move to the “Speed Adjustment” process.

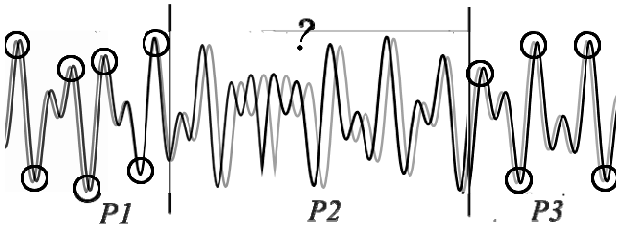


Figure 12. Coinciding Unknown Clip and Its Most Matched Clip

6) Speed Adjustment: Sometimes songs may broadcast with a small playback speed variation, either slower speed or higher speed than the normal speed because of some faults of playing device. Listener’s auditory system may not be sensitive enough to hear these speed differences. But this may cause to decrease the overall system accuracy.

We solve this problem using some speed adjustment algorithm. We can match unknown clip against the database songs by changing the speed. After doing some experiments we found specific speed range that that can practically happen. We change the playback speed under maximum +-5% limit. We do this step by step and in each step we try to find matching song. If a matching song is found then we stop the process before reaching the maximum limit.

7) Generate Report: This is the final process of our system. We already have sets of matched song ids. Therefore generating a report is quite easy. We are not going to discuss this process in detail. We have provided some sort of facility to generate reports via internet. But to do this user should have permission or authentication to access data.

V. EVALUATIONS

In this section we evaluate our implementation and analyze the obtained results. To evaluate the system properly, we should insert considerable amount of songs into the database. Otherwise result may not accurate due to absence of variety of songs. As the first step, we take collection of songs which belongs to various genres. Finally we inserted 1200 songs considering following factors.

- Different genres such as classical, POP, Rock, Jazz and so on
- Combinations of singers like male, female and duets
- Songs with different energies
- Different languages like Sinhala, English and Hindi
- Different ages of singers

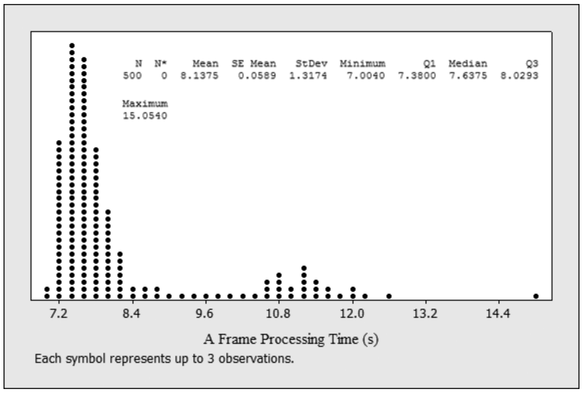
After adding 1200 songs, our database consist of 2,267,329 fingerprints which is not a small number. Therefore we will get very fair and unbiased results from our experiments.

We divide our evaluation into major three parts which are performance evaluation, accuracy evaluation and song extraction accuracy evaluation. We execute all the test cases on a Desktop computer with this configuration. Os: Windows7 (64 bit), RAM: 4 GB, Processor: Intel core i3, Processor Speed: 3.1 GHz.

A. Performance Evaluation

This is a real time system. It means we process audio clips while they are broadcasting. Therefore execution time is really important factor in this case. As we discussed in the implementation, we buffer five minutes long audio clip before it is conveyed to process. While first clip is processing, next audio clip is buffered simultaneously. According to this scenario, we can't reduce the buffering time (it takes obviously five minutes). Our aim was to process each five minutes long audio clip within next five minutes. If we can achieve this then we can say that our system performance is at optimal level.

As we already know, we always process 40 seconds long audio clips. Here we observe processing times of 500 frames. Then we draw dot-plot using these time values, refer the Figure 13. According to this figure, mean processing time is 8.1375 seconds. It is very small value and we can say that our system approximately takes only one fourth of the duration of a particular clip for the processing.



N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median	Q3
500	0	8.1375	0.0589	1.3174	7.0040	7.3800	7.6375	8.0293

Maximum
15.0540

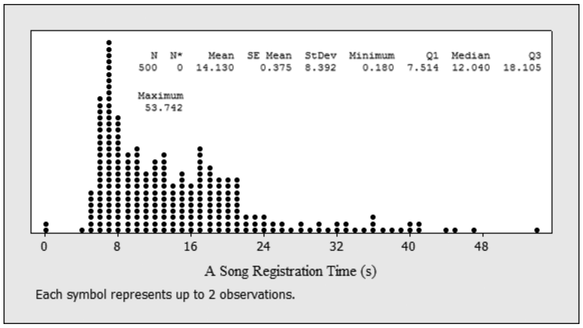
Each symbol represents up to 3 observations.

Figure 13. The distribution of frames processing time in seconds and important statistical results.

When we talk about the execution speed, time taken to add a song to the database is another important value. Therefore we observed song insertion (or registration) time by adding 500 songs into the data base. Then we prepare a dot-plot using obtained results, refer the Figure 14. Average song registration time of these songs is 14.13 seconds. However, this value is depended on the duration of the song.

B. Accuracy Evaluation

In this section we will evaluate the accuracy of our system. To do this we did the following tests. Accuracy evaluation is not a trivial task for this kind of system since radio channel can broadcast anything with any combination. We have to consider all of these scenarios since system reactions may depend on these scenarios.



N	N*	Mean	SE Mean	StDev	Minimum	Q1	Median	Q3
500	0	14.130	0.375	8.392	0.180	7.514	12.040	18.105

Maximum
53.742

Each symbol represents up to 2 observations.

Figure 14. The Distribution of Songs Registration Time in Seconds and Related Statistical Results

Therefore we prepare several test cases which are commonly occurred on a radio channels. Then we collected considerable amount of examples for each case. After that we observed what the system respond for these cases. Let's consider only major test cases one by one with statistical representations.

- *Test Case 1(General Case):*

In this test case we tested the system for normal song with completed, high quality without any noisy, commercials, water marks. We took twenty random songs which are already in the database and twenty other songs which are not currently in the database. Then execute our system for this test case.

System provides three kinds of outputs for this case. System can identify an unknown song as the correct one or system can totally omit one or more unknown song (without matching correctly or incorrectly) or system can introduce completely new one or more song(s) as matched ones. We have achieved 97.56% success in this general case as well as 2.44% error rate which was happened due to incorrect match. System could identified all exist songs correctly as well as not exist songs, were totally omitted without matching with any existing songs. But system introduced a completely new song as a match therefore it is an error, refer the Figure 15.

- *Test Case 2(Optimal Duration of Audio Clip)*:

In this case we tested to identify ”required minimum broadcasting duration” of a song to be matched correctly. Again we selected ten random songs which are currently in the database and ten other songs which are not currently in the database. Then we extract 10 seconds from each and every songs, 15 seconds from each and every songs, 20 seconds, 25 seconds and so on up to 40 seconds.

After this, we tested our system for each and every scenario. Results are shown in the Figure 16. According to the Figure 16, it should play more than 35 seconds to identify a clip correctly. The detection rate is decreased gradually when playing duration is decreased. Therefore we select 40 seconds as the optimal duration for the framing process.

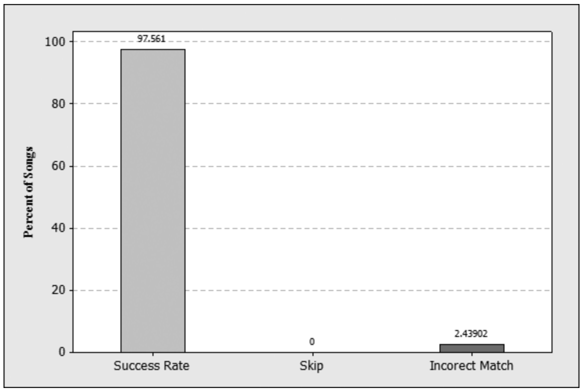


Figure 15. Results distribution of the general case (Test Case 1)

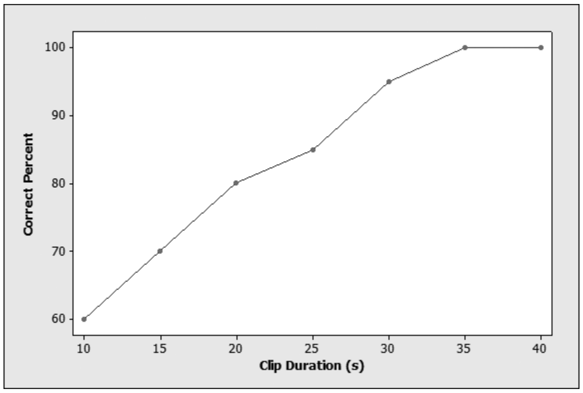


Figure 16. Song detection percentage against the clip durations(s)

- *Test Case 3 (Sequence of Songs from Same Singer)*:

In this test case we tested the system behavior when more than one song is playing as a stream without any silent in between two songs. Most of the time we feel that songs which are sang by the same singer are similar. Therefore there is a high probability to match a song with incorrect one of the same singer.

This probability may increase when two songs from the same singer fall in to the same frame. Therefore we select ten singers and three songs from each singer which are currently in the database. Then these three songs are appended one after the other so that we obtained 10 clips. Also we create similar data set for songs which are not currently in the database. Ultimately we obtained 60 songs, and then we direct all these songs to the system. Obtained results are shown in the Figure 17.

According to the Figure 17, we have achieved 87.10% success and 12.9% error. As we mentioned earlier, when two songs fall into a single frame then it may detected as incorrect song. Fortunately these kinds of cases are very rare in real broadcasting environment.

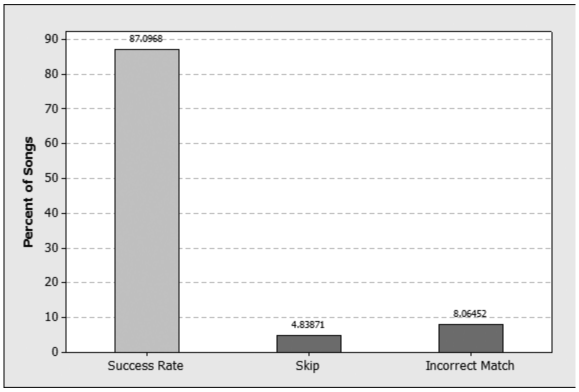


Figure 17. Three Songs from a Singer are Joined One after the other. Result distribution against this case.

We did the same test for different singers instead of same singer as well. Two songs can be mapped to the same frame. As above test, we prepared 60 songs and tested the system for this test case. In this case, we have achieved 90.16% success and 9.84% error. However this is also a rare case.

- *Test Case 4 (Test for Non-song Objects)*:

We tested the system behavior for non-song object such as dramas, commercials, talks, discussions and so on. We recorded actual non- song object from radio channels then tested the system on this audio clips. We directed 60 frames to the system. The system did not match any frame with a song in the database. All frames were skipped, and therefore we achieved 100% accuracy in this case.

- *Test Case 5 (Test for Position Independency)*:

In this test case we tested two things. The first is to test the system behavior when a part of a song is presented and the second test is to test how the system behaves when we present a part of a song with position independent. Radio channel can play a part of a song which can be extracted from anywhere of the song. To test this, we extract 100 seconds long clip from a random position of the song. Again we extract ten, 100 seconds long clips from songs which are currently in the database and ten other songs having 100 seconds long clips from songs which are currently not in the database. Then these 20 cases are directed to the system. Obtained results are shown in the Figure 18.

- *Test Case 6 (Adding Commercials at the Middle of Song)*:

When we are listening to a radio channel you can hear playing some commercials in the middle of a song. During the song, first they stop the currently playing song and then play some commercial for a short time and then continue the same song again. In this test case, we observe the system behavior against these kinds of scenarios. As the previous test cases, we select 10 such songs currently in the database and 10 other songs which are not currently in the database. Then we tested the system on these cases. Figure 19 shows the obtained results.

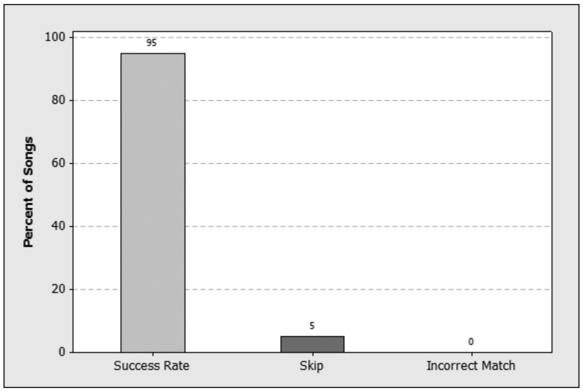


Figure 18. Behavior of the system against position independent short time(100 s) clips

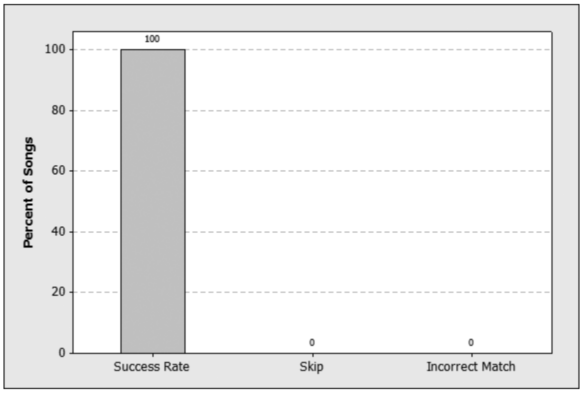


Figure 19. When playing commercials at the middle of the song

- *Test Case 7 (Songs with Background Watermark)*:

Usually radio channels add some commercials or talks as watermarks. In this test case song is not stopped while commercials are also playing in the background. There are major two kinds of watermarks. First one is that volume of the song is reduced up to some level so that the emphasis on the commercial. The other one is that song and commercial are played at the normal volume but we can hear song and commercial separately. Normally the second one is used for short time commercials. In this test case, we take 10 samples from the first scenario and 10 from the second scenario. We achieved 100% accuracy for this case as well. We can conclude that the background watermarks are not a problem at all for our system. Actually our framing module helps us very much in this case.

- *Test Case 8 (Songs Destroyed by Noisy)*:

This is the most important and the most probable test case. When we are listening to a radio, hearing noises is obvious due to several factors. Noises can be added to the radio channel due to improper channel frequency tuning, environmental disturbances, weak radio signal strength and so on. In this test case we test the noisy level which can be handled by the system. We generate different levels of continuous noises and added them to the tested song. First we generate a noisy with the amplitude level of 0.1, then it is mixed with 10 songs which are already in the database and 10 other songs which are not already in the database. This process is repeated by changing the noisy level from 0.10 to 0.50. Ultimately we generate 180 noisy destroyed songs then we tested the system on this test data. Obtained accuracy level is shown in the Figure 20.

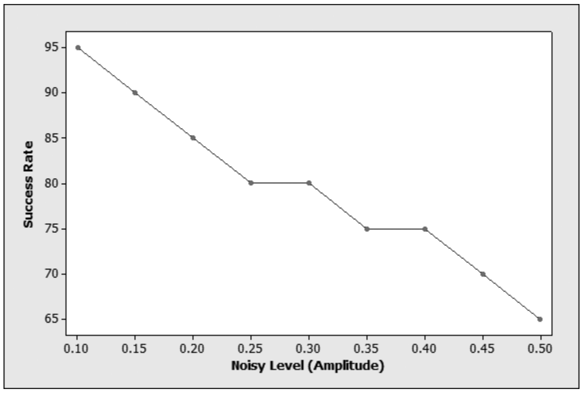


Figure 20. Accuracy of Noisy Destroyed Songs

We have achieved higher accuracy level even if song is completely destroyed by the noisy with the amplitude level of 0.5. However, noisy is not held throughout the song when we come to real situation. Therefore we can expect higher accuracy rate than this.

C. Song Extraction Accuracy Evaluation

We already know that this module is used to separate song object from the others. To evaluate this module we first extract only pure non-song objects such as commercials, talks, conversations, dramas, educational programs like quizzes, news and so on from real broadcasted audio stream. We prepared 300 frames for this (i.e. approximately 3.33 hours long audio stream). Thereafter we directed these frames to the song extraction module. This module can identify each frame as a non-song object or as a song object. We do the same to actual song objects. It is really easy since we can take sets of various songs and test the module on this data. We test the system using 800 frames. It is equal to 150 songs. Behavior of the module is shown in the Figure 21.

Our system can remove 57% non-song object correctly. Approximately system resources are not wasted if the frame is a non-song object. Therefore this module can speed up the system by 57% as well as reduce the unnecessary utilizations of system resources. This is a considerable advantage provided to our system by this module.

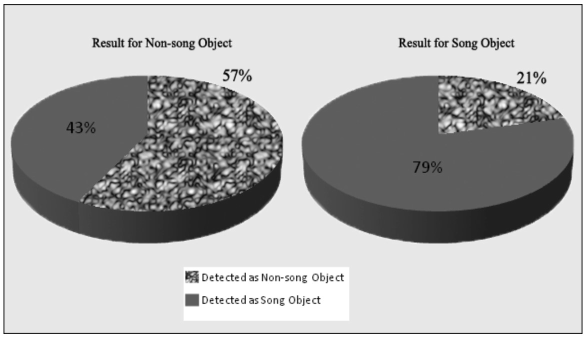


Figure 21. Results distribution of song extraction module

D. Summary of the System Accuracy

We tested our system against the various kinds of test cases. To get an overall idea about the system accuracy, we can summarize the obtained results. Table I shows the summary of the obtained results.

Table I
OVERALL SUMMARY OF THE TEST CASES

Test Case	Error Rate (%)	Success Rate (%)
General Case	2.439	97.561
Sequence of Songs from the Same Singer	12.903	87.097
Sequence of Songs from Different Singers	9.836	90.164
Continuous Non-song Objects	0	100
Short Position Independent Clips	5	95
Adding Commercials at the Middle of Song	0	100
Songs with Background Watermark	0	100
Noisy Destroyed Songs	5	95
Different Songs with the Same Melody	0	100
Overall Accuracy Level	3.909	96.091

VI. CONCLUSION

Let’s recall the problem which we tried to take in hand through- out this research. Radio broadcast monitoring is a still manual process, especially in developing countries. At the end of this manual process, they generate some sort of reports including the details of broadcasted song history. Thereafter the owners of the songs will be paid accordingly. This is a tedious and difficult task.

We provide a highly robust audio monitoring approach for radio broadcasting as the solution for the proposed research problem here. Our proposed approach achieved overall 96% accuracy in recognition of songs broadcast over radio channels. Thus, system can successfully handle almost all real situations which occur in an actual radio broadcasting environment. We prove that capability in the section V. Our approach does not require any additional hardware. Thus, we can conclude that our approach can be used to monitor radio broadcasting channels with high accuracy, high efficiency and low cost.

VII. FUTURE WORKS

We can suggest that there is a room for greater improvement of our system and other systems in general. For instance, radio station can play a song with different pitch, tempo, with or without echo and so on. In future, our research can be extend to identify different versions of same song from the copyright point of view. Therefore, we can extend our research in order to classify these versions of the same song into the same class.

We used relational database to store hashes and other related data. But we can't maintain RDB when database grows. So we have to use some novel approach like distributed database in order to cope with this problem. Further, we can classify songs based on genre into sets of groups. This will reduce the search space. So we can join our research with some genre classification approach to achieve better results.

ACKNOWLEDGMENT

I offer my sincerest gratitude to my supervisor Dr. Lakshman Jayaratne, who has supported me throughout my research. I would like to thank Mr. Dulan Watugala for motivating and coordinating the project. I would also like to show my gratitude to Mr. Brian Wijesuriya for supporting me. Finally thank everybody who contributed to the successful realization of my project.

REFERENCES

[1] J. Haitsma and T. Kalker, “A highly robust audio fingerprinting system,” in Proc Int Symp Music Info Retrieval (M. Fingerhut, ed.), vol. 32, pp. 107–115, Ircam - Centre Pompidou, 2002.

[2] M. Miller, M. Rodriguez, and I. Cox, “Audio fingerprinting: Nearest neighbor search in high dimensional binary spaces,” The Journal of VLSI Signal ProcessingSystems for Signal Image and Video Technology, vol. 41, no. 3, pp. 182–185, 2002.

[3] G. Reynolds, D. Barry, T. Burke, and E. Coyle, “Towards a personal automatic music playlist generation alogorithm: the need for contextual information,” in Conference papers, p. 11, 2007.

[4] A. de Almeida Guimarães, “The rds (radio data system) as a transmission way of information for automotive vehicles,” 2000.

[5] B. Denby, O. Romain, and S. Hariti, “A software radio approach to commercial fm content indexing,” in 11th International Workshop on Systems, Signals and Image Processing, IWSSIP, vol. 4, pp. 13–15, 2004.

[6] D. Kopitz and B. Marks, RDS: the radio data system. Artech House, 1999.

[7] Cox, M. Miller, and J. Bloom, “Watermarking applications and their properties,” in Information Technology: Coding and Computing, 2000. Proceedings. International Conference on, pp. 6–10, IEEE, 2000.

[8] H. Neuschmied, H. Mayer, and E. Batlle, “Content-based identification of audio titles on the internet,” in Web Delivering of Music, 2001. Proceedings. First International Conference on, pp. 96–100, IEEE, 2001.

[9] P. Cano, E. Batlle, H. Mayer, and H. Neuschmied, “Robust sound modeling for song detection in broadcast audio,” Proc. AES 112th Int. Conv, pp. 1–7, 2002.

[10] P. Cano, E. Batlle, T. Kalker, and J. Haitsma, “A review of audio fingerprinting,” The Journal of VLSI Signal Processing, vol. 41, no. 3, pp. 271–284, 2005.

[11] A. Wang, “An industrial strength audio search algorithm,” in International Conference on Music Information Retrieval (ISMIR), vol. 2, 2003.

Nishan Senevirathna is currently a fourth year computer science undergraduate at University of Colombo School of Computing, Sri Lanka. His research interests include Multimedia computing, Image processing, High performance computing and Human computer interaction.

Dr. Lakshman Jayaratne - (Ph.D. (UWS), B.Sc. (SL), MACS, MCS(SL), MIEEE) obtained his B.Sc (Hons) in Computer Science from the University of Colombo, Sri Lanka in 1992. He obtained his PhD degree in Information Technology in 2006 from the University of Western Sydney, Sydney, Australia. He is working as a Senior Lecturer at the University of Colombo School of Computing (UCSC), University of Colombo. He has wide experience in actively engaging in IT consultancies for public and private sector organizations in Sri Lanka. At present, he is working as a Research Advisor to Ministry of Defense, Sri Lanka. Also he is the present President of Chapter of Sri Lanka, IEEE. His research interest includes Multimedia Information Management, Multimedia Databases, Intelligent Human-Web Interaction, Web Information Management and Retrieval, and Web Search Optimization.