CHAPTER 7

MUSIC INFORMATION RETRIEVAL

Using the music and non-music components extracted, as described in chapters 5 and 6, we can design an effective Music Information Retrieval system. In this era of multimedia, a large collection of digital music has become available (Tzanetakis 2003). Content based music retrieval in the context of Western music has been proposed by various researchers (Foote et al 2002) (Peeters et al 2002). In the context of Indian music, Content based music retrieval based on Indian Classical music was proposed by Chordia et al (2009). This necessitates the efficient management and retrieval of music collection, from large music databases available or extracted from the Web. Therefore we used indexing for Music Information Retrieval, as an application to test the effectiveness of the features extracted. This chapter explains indexing schemes that could exploit these features.

7.1 OVERVIEW OF MUSIC INFORMATION RETRIEVAL

In general, Music Information Retrieval is performed either by using manually annotated metadata, or by using the content features of the input. In the Metadata based retrieval, similar pieces of music are retrieved, when the input query is submitted in terms of text information, like the title, Singer or Genre. This is typically performed by manually indexing the musical pieces with information on the Genre, Singer, Lyricist, etc., which are later used as metadata for the process of retrieval.
Content-based retrieval has gained more attention, compared to mere metadata based retrieval. The choice of features and representation of the content is critical to retrieve relevant information (Jiau and Chang 2008). Content-based music retrieval is defined as the process of finding occurrences from a music corpus, which are similar to an input segment of music. The input query for content-based retrieval is typically specified in terms of either a humming or an example musical piece. In the case of Query by humming, a tune is hummed from which features are extracted, which are used to match with the features of music in the database to retrieve the relevant music. On the other hand, in the case of Query by example, the input as a whole is considered for matching (Tsai and Wang 2004, Itoyama et al 2010, Itoyama et al 2009, Thoshkhana and Ramakrishnan 2005).

The three basic blocks of any information retrieval system are, query processing, indexing and searching, and this is applicable to a MIR system also. The query in this case is a music file, which is processed using signal processing techniques, to extract temporal, spectral or Cepstral metadata features, as explained in Chapter 2. From these features the music components are determined, and these are used as the key values for the indexing module of the information retrieval system.

Content-based music indexing is performed either manually or automatically. Manual indexing refers to indexing the musical piece, based on the title, Singer, Genre, query phrase etc., which have to be manually provided, and hence, this is a cumbersome process. For automatic indexing of songs, in addition to the manually provided information, such as the title, film details (in case of film music), etc., some features like the Singer and Genre are automatically discovered. In addition, in order to provide more fine grained information about the musical piece, raw signal level features, such as the Mel-Frequency Cepstral Coefficient, spectral flux and centroid or a
combination of other features, are used for indexing. Additionally, these raw signal level features can be used to extract musical features like the rhythm, melody, timbre, intensity and melody string, which can also be used for indexing (Helen and Virtanen 2010, Helen and Virtanen 2007, Helen and Lahti 2006). This automatic indexing of songs based on the metadata, raw signal level features and musical features, allows the retrieval of music based on different perspectives.

In this thesis, the components of music that have been identified as described in the previous chapters, have been used for indexing and retrieving music files from a music corpus. In addition, a new indexing technique that exploits these components has been proposed for the Music information retrieval system. In order to show the performance of the indexing technique, we have built an information retrieval system that uses the existing algorithms for the other stages of retrieval, except query processing, where a minor modification has been carried out.

7.2 MIR FOR WESTERN MUSIC

Content based music information retrieval was proposed that cater to the various blocks of query processing, indexing and retrieval by researchers. Several algorithms have been proposed by various researchers for the process of music information retrieval. As discussed in Chapter 2, a scalable P2P system for content based music information retrieval has been developed by Tzanetakis (2003), using the Short Time Fourier Transform and Mel-Frequency Cepstral Coefficients, to represent the sound texture, rhythm and pitch content to be used as the parameters for retrieval.

It is evident, that automatic content based music information retrieval relies greatly on a good indexing algorithm. The typical
characteristics of an efficient indexing algorithm are efficiency in time and space, and good precision and recall.

In an algorithm proposed by Rauber for automatic indexing of music, the Genre of the music is used for indexing either directly, or through features conveying the Genre of the musical piece (2002). In another work by Shih et al, the authors have used the tempo characteristics for indexing (2001). Yang used a spectral indexing algorithm, based on multiple hash tables as a mode of indexing, and then retrieval is performed based on the Hough transform (2002).

Doraisamy and Ruger have proposed a music information retrieval system that uses an N-gram approach for indexing (2003). Another N-gram based indexing technique using MFCC features was proposed for Indonesian Folk Songs by Marsye and Adriani (2009).

Lo and Wang have proposed another work for indexing, using suffix trees and addressed the problem of the variable query length of the music segment during retrieval (2009). The authors have used a concept called Grid Suffix trees for the construct of indices which uses multiple features for music indexing (Lo and Wang 2009). In another grid-like structure called dual-ternary indexing, the authors have used a two-dimensional grid, and three number notations to represent the pitch content (Jiau and Chang 2008).

In general, the various indexing techniques that are normally used for multimedia data, are the suffix tree, hash table, multi-dimensional suffix tree, N-gram model or some variations of these multi-dimensional representations. In our work we have identified a number of features from the input music, and hence, a pattern based suffix tree representation cannot be used. After analyzing various music indexing techniques, we have proposed a
modification to the basic dual-ternary technique (Jiau and Chang 2008) by exploiting the Carnatic music characteristics.

We have concluded that for indexing music we require a multi-dimensional representation, such as a variation of the hash table for indexing that accommodates all the features and the identified components, referred to as the Modified dual-ternary indexing algorithm. This algorithm was not time efficient since the number of features involved was large, and moreover this technique requires a phrase contour computation. Therefore, we proposed one more structure for indexing, which is again based on the Hash-table concept, which we call the Multi-key Hash-table structure. The algorithm is based on exploiting the collision resolution of the hash structure. This algorithm has been tested for Carnatic music songs, and the performance is compared with a modified dual-ternary indexing algorithm.

### 7.3 MODIFICATION TO DUAL-TERNARY INDEXING TO SUIT CARNATIC MUSIC

In the dual ternary indexing approach proposed by Jiau and Chang (2008), the songs stored in the database are segmented into meaningful phrases and indexed. This approach uses a two-dimensional array for indexing, and the two indices that are used are called $I_{PRE}$ and $I_{SUF}$. $I_{PRE}$ and $I_{SUF}$ are calculated from the prefix and suffix part of the phrase contour as given in Equation (7.1) and (7.2).

\[
I_{PRE} = f(P_{PRE}) = \sum_{i=0}^{i=L_{PRE}-1} PC_{i+1-L_{PRE}} \times 3^{L_{PRE}-1-i} \tag{7.1}
\]

\[
I_{SUF} = f(P_{SUF}) = \sum_{i=0}^{i=L_{SUF}-1} PC_{i} \times 3^{L_{SUF}-1-i} \tag{7.2}
\]
The total length of the index is fixed prior to the implementation. When collision occurs, it is resolved by open chaining. The algorithm computes a phrase contour, which when hashed to the same indices, is stored in the same slot of the indexing structure. Figure 7.1 shows the dual ternary indexing structure having a two dimensional matrix structure. The phrase contour consists of three values ‘0’, ‘1’ and ‘2’. A ‘0’ is assigned to a segment if the frequency of the current segment is the same as the previous one, a ‘1’ if the frequency of the current segment is less than that of the previous segment and a ‘2’ if the frequency of the current segment is greater than that of the previous segment.

In our work, the index consists of the swara components. We have made a modification to the ternary indexing in the way that this contour is determined. We initially determine the tonic of the signal using the algorithm specified in Chapter 4. Then, we determine the dominant frequency of every small segment, and determine the swara sequence S, R, G, M, P, D, and N as explained in Chapter 4.

After determining the swara sequence, we represent the phrase using ‘0’, ‘1’ or ‘2’ indicating the change in the swara between the successive notes. In figure 7.1, phrase1 and phrase2 may have the same contour but may have a different melody string or swara pattern, corresponding to different Ragas or may belong to different music files.

![Figure 7.1 Dual Ternary Indexing Structure](image-url)
In addition to the modification done during indexing, we have modified the algorithm during the query processing stage. In the dual-ternary algorithm proposed by Jiau and Chang (2008), the aim of matching performed during query processing, was to retrieve all phrases that have both the same prefix and suffix as those of the query. The aim of similarity matching was to retrieve all phrases that had either the same prefix or suffix as that of the query, but differed in length. Here the musical piece can be easily retrieved, irrespective of the index length by calculating $I_{\text{PRE}}$ and $I_{\text{SUF}}$. If the query is available in the prefix part of the phrase, then the suffix query expansion is done to generate all possible queries, which have the same prefix as the input query. Similarly, if the query is the suffix part of the phrase, then the prefix query expansion is done. Thus, entries in the same row or column are retrieved for similarity matching. However, if the query is between the prefix and the suffix, then retrieval would be difficult. In addition to that, in the dual-ternary algorithm, songs are stored in the database by segmenting them into meaningful phrases. The algorithm is based on the assumption that users tend to submit a perceptually meaningful query phrase. It uses the representative music fragment extraction technique (Chang and Jiau 2004) to extract the representative melody fragments. We observed that the dual ternary algorithm will not be able to handle segments obtained using other segmentation techniques, where every segment need not correspond to either the prefix or the suffix of a melody contour to be stored in the indexing table. For example, if the stored melody contour $S = (2, 2, 0, 0, 1, 2)$ and Query $Q = (2, 0, 0)$, then $S$ will not be retrieved though it contains $Q$.

In order to avoid this dependency on one segmentation technique and the necessity of the query phrase being either the prefix or the suffix of the signal, we modify the query processing stage of the existing dual ternary algorithm, by using a sliding window technique to handle the query (Chang
and Jiau 2004a). We handle the query phrase by using a fixed size sliding window, so that any part of the query could match with the melody contour. If \( N \) is the length of the melody contour and \( W \) is the window size, then the number of sub-segments is \( N-W+1 \). For example, if \( N = 8 \) and \( W = 4 \), then the number of sub-segments \( = 8-4+1 = 5 \). Since we have considered this algorithm for Carnatic music, we have used the sliding window size of 7 to accommodate the 7 distinct swaras that can occur in a sequence. This is the size of the length of the swara in a Parent Raga definition, discussed in Chapter 1. Using this sliding window technique, we compute the \( I_{\text{PRE}} \) and \( I_{\text{SUF}} \) of the songs, and use these values to index the songs. Retrieval is performed in a manner similar to that of the original algorithm of dual ternary indexing, except for the query processing (Lo and Wang 2008).

### 7.4 MULTI-KEY HASHING INDEXING ALGORITHM

The modified dual-ternary algorithm discussed above incorporates only the swara pattern in the index. According to the literature for indexing, the retrieval algorithms for Western music had given better results when more signal level features were used. In addition, we would also like to explore the use of the multi-dimensional grid structure for efficient storage, which motivated us to propose this algorithm, which is a combination of hashing and a grid structure.

In order to accommodate the rich features in music such as the raw signal level features, namely, the Spectral Flux, Spectral Centroid, Tonic, MFCC, CICC and music components extracted from these features like Raga and Singer, we have designed a new index structure based on the Hash tables (Aho et al 1983). Multi-key hashing is similar to hashing in which we use a key to compute the index using a hash function, and store this element in the hash table. In our application, the various characteristics that have been
identified from the music namely, Raga, and Singer, are some of the keys. In addition, we have also used the signal level feature values, namely, the Tonic, CICC, Spectral centroid, and Spectral flux as keys in the Hash structure. The choice of the Tonic is necessary to retrieve songs by a Singer(s) in a particular pitch. The other signal level features are typically used for the process of similarity match, when the exact match is not available. Queries such as similar Ragas, similar Singers, etc can be handled since we have incorporated fine grained features about these music components.

We have proposed an algorithm that uses a combination of chaining and a variation of re-hashing, which we call as Multi-key hashing. The design of the hash structure used by the algorithm is described in Figure 7.2.

Figure 7.2 Multi-key Hashing Structure

In the above structure, Element1, Element 2, and Element 3 correspond to the available ‘n’ data elements that need to be indexed. The Raga, the Singer and the tonic are used to identify each element of the input music. In addition to these characteristics, the signal level features are also used to identify each of these ‘N’ elements. Let us represent 1(k1) as the 1st key of the 1st element, 1(k2) corresponds to the 2nd key of the 1st element and
so on. Using the first key, the index is computed. If there is no collision, the element is added to the hash table. In the event of a collision we chain the element and then use the same hash function, to compute the index using another key. This is illustrated in Figure 7.2. $F(1(k1))$ indicates that $F$ is computed using the first key of the first element which is actually mapped onto an index value.

The Multi-key hashing algorithm for index computation is similar to a simple hash structure when there is no collision. An index is computed using the hash function, and the data is stored in that index. In the event of collision, we first chain the element in the index, and in addition, we use the next key of the data and the same hash function to re-compute the index to have a copy of this data. The element is stored in this index if there is no collision. If collision occurs then we use the next key to compute the index. Thus, one single data can occur in multiple places if the key has resulted in a collision of the index value. The order in which the key should be chosen from the list of available keys is determined, based on their uniqueness and arranged in the decreasing order of robustness. Thus the key, which is most robust, is ranked first followed by the next robust key and so on. During retrieval, the same process is carried out and even though we use only one hash function, the process of chaining allows duplication.
The multi-key hashing algorithm is given below:

```
Multi-keyHashing (Input dataelement n)
{
  For elements = 1 to n
  Extract-feature(element[i])
  Key[] = Music components and features
  For dataelement 1 to n
  {
    while (key)
    {
      Index = Hashfunction(key)
      if (!Collision)
        Store element in the Index
        Go to next element
        Break;
      Else
        Chain (element)
        Choose next key;
    }
  }
}
```

In our work, we have considered this algorithm for indexing Carnatic music songs, using our extracted signal level features and identified components. For our work we have used spectral parameters, in which the frequency is mapped to a human perceived swara representation, the Spectral flux, Spectral Centroid and Mel-frequency Cepstral coefficients (MFCC), and the CICC. The swaras that are derived from the input song using the tonic of the input signal and other frequencies of the input, as explained in Chapter 4, are mapped to the Raga and the characteristic phrase is determined. This characteristic phrase, called a melody string is important because Carnatic music is essentially characterized by the tonic dependent swara phrase, and
hence, this is the first level key used for indexing. In addition to this phrase, other components like Raga, Singer, Tonic, spectral flux, spectral centroid and CICC are used as the second level key for indexing the musical piece. Later these features are used in the same order, to compute the index of the hash table of the query to perform retrieval. This multi level index allows the retrieval of music for queries that incorporate music level indicators such as the Raga, Singer, tonic etc. An overall description of our MIR system is shown in Figure 7.3.

![Figure 7.3 General MIR](image)

### 7.4.1 Results and Analysis

In this work, we considered 1500 songs from Tamil films and Carnatic music sung by 15 Singers as discussed in Chapter 6. The input query is a song lasting 20 seconds. This length of the query was varied for analyzing the performance of the algorithms for varying query lengths. The input songs are polyphonic, and hence, consist of voice and accompanying Instruments. After determining the features and the components as discussed in Chapters 4, 5, 6, we estimate the frequency based phrase contour, by comparing the
frequency segments of the successive frames used by the Modified Dual ternary indexing algorithm. In the other approach, the index is constructed using the Multi-key hash structure with features, like the swara phrase, Raga, Singer, Tonic, spectral flux, spectral centroid and the CICC.

As already discussed, a sample of 1500 songs belonging to different Singers is taken as the training data for feature extraction and component identification. Two different databases are populated with the indices computed for 2500 songs, using Modified Dual Ternary and Multi key hashing algorithms. The observations are tabulated in Table 7.1.

Table 7.1 Retrieval time and space efficiency of the indexing algorithms

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Indexing algorithm used</th>
<th>Average Time Efficiency for similarity matching (seconds)</th>
<th>Average Time efficiency for Exact matching (seconds)</th>
<th>Space Efficiency (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dual – ternary with modification</td>
<td>4.568</td>
<td>3.718</td>
<td>333 (5_5)</td>
</tr>
<tr>
<td>2</td>
<td>Multi-key Hashing</td>
<td>3.857</td>
<td>0.049</td>
<td>1280</td>
</tr>
</tbody>
</table>

The observations are tabulated based upon choosing songs belonging to 15 Singers in 25 different Ragas, and in varying Genres. The characteristics of the Raga or Singer are identified along with the associated features and used for indexing. From Table 7.1 it is clear that Multi-key hashing has good time efficiency for retrieval, than modified dual-ternary indexing, when an empirical analysis is done on a data set consisting of 1000 songs in a database of 2500 songs. However the space efficiency of the dual-ternary indexing algorithm is better compared to the Multi-key hashing algorithm. For the modified dual ternary indexing algorithm, we implemented a 2-dimensional table of 5 * 5 to compute the \( I_{pre} \) and \( I_{sur} \), and observed that on
an average the space complexity is constant. On the other hand, since Multi-key hashing saves multiple copies of the same element in different positions, it had a higher space complexity. However, the time-complexity of Multi-key hashing is better, due to the fact that it uses a hash structure, and typically, hash algorithms are $O(1)$, while on the other hand, in the modified dual-ternary indexing, the concept of computing the phrase contour results in increased time complexity.

To trade off between time and space, we analysed the performance of these two algorithms for the time taken to retrieve, in the event of a similarity match and the exact match. The results are shown in Figures 7.4 and 7.5 respectively. In the algorithm for determining Multi-key hashing, we used the Raga, Singer, CICC, Tonic, and the Spectral Centroid as keys in the specified order.

![Figure 7.4 Performance for different query length for similarity matching](image_url)

**Figure 7.4 Performance for different query length for similarity matching**

As can be seen, for similarity matching based retrieval, the Multi-key hashing algorithm had a good time complexity compared to the Modified Dual-ternary algorithm. The similarity we are comparing could be in terms of
the features of Raga, Singer, CICC, Spectral Centroid, and Tonic, which were
used as keys to compute the index. However, the Modified Dual-ternary
considered the similarity in terms of the varying swara pattern only, and
hence, the similarity was measured in terms of these fluctuations in the swara
of the input. The computation of the phrase contour is required in the
Modified Dual ternary indexing algorithm, which is the reason behind the
increase in time during retrieval when compared to Multi-key hashing. The
Multi-key hashing algorithm avoided the increase in time due to linear
chaining by duplicating the data in multiple positions. However, both the
algorithms had an increase in time for retrieval when the length of the query is
increased.

The same performance analysis on varying query lengths is
performed for exact matching, and the observations are shown in Figure 7.5.

![Figure 7.5 Performance for different query length for exact matching](image)

**Figure 7.5 Performance for different query length for exact matching**

Exact matching is referred to as one in which the retrieval
algorithm was able to match the information in the query with the exact data
in the database. In the case of Multi-key hashing, the exact match of the Raga,
and Singer is attempted by the algorithm rather than the Tonic or other signal level features, and hence, the time taken for retrieval is quite less. When the query length is small, the algorithm essentially identified the Singer rather than the Raga for performing an exact match. On the other hand, the Modified Dual-ternary algorithm tried to match the fluctuations in the swara phrase, during both the similarity and the exact match. Hence, the time taken to retrieve is high compared to the Multi-key Hashing algorithm.

The Modified Dual-ternary indexing algorithm used the variations in the swara as a key to compute the index, whereas the Multi-key hashing used the components of music for indexing. Therefore, when a query is very specific, then the Multi-key hashing algorithm performed better than the Modified dual-ternary indexing. The results are shown in Figure 7.6.

![Figure 7.6 Relevant Retrieval comparison of the indexing algorithms](image)

**Figure 7.6 Relevant Retrieval comparison of the indexing algorithms**

It can be observed from Figure 7.6, that when the input query was a Raga, the Modified Dual-ternary algorithm always tried to extract the varying swara patterns from the input for matching. This algorithm was able to
retrieve the same Raga in 68% of the instances, but for the remaining 32% the information that was retrieved was totally tangential. The feature rich Multi-key hashing algorithm, however, retrieved 75% of the instances with the same Raga. For the other 10% the algorithm retrieved its Child Raga (approximate match), and the remaining 15% of the instances was retrieved based on other features, and had some similarity to the given input Raga. However, when the input query was a Singer, the Modified Dual-ternary algorithm retrieved 65% correctly; for the remaining 35% the retrieval was based again on the varying swara pattern of the query, and not on the Singer characteristics. On the other hand, the Multi-key hashing algorithm retrieved songs of the same Singers 80% of the time, while for the remaining 10%, the songs of Singers who have almost the same Tonic, were retrieved. However, for similarity matching, both the algorithms were comparable.

We then estimated the Precision and Recall measure to test the performance of the algorithm. The results are tabulated in Table 7.2.

### Table 7.2 Precision and Recall

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Modified Dual Ternary Indexing</th>
<th>Multi-key Hashing Indexing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>69%</td>
<td>65%</td>
</tr>
<tr>
<td>Recall</td>
<td>68%</td>
<td>73%</td>
</tr>
</tbody>
</table>

The precision rate of Multi-key hashing is lower than that of the modified dual ternary indexing. This difference is because of the availability of one element in multiple positions, and hence, the number of songs retrieved is more than what actually should be retrieved. On the other hand, the recall of Multi-key indexing is high, again because of the presence of elements at multiple locations and hence, whatever should be retrieved was retrieved. The comparison of the two algorithms is given in Table 7.3.
**Table 7.3 Comparison of the two indexing algorithms**

<table>
<thead>
<tr>
<th>Modified Dual-Ternary Indexing</th>
<th>Multi-key Hashing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses contour representation of the melody string as the key for indexing</td>
<td>Uses the melody string, Raga, Singer, MFCC, Flux as keys for indexing</td>
</tr>
<tr>
<td>Resolves collision by chaining</td>
<td>Resolves collision by chaining and duplication</td>
</tr>
<tr>
<td>Compromises time over space complexity</td>
<td>Compromises space over time complexity</td>
</tr>
<tr>
<td>Suitable for Western music</td>
<td>Suitable for Western music and Carnatic music</td>
</tr>
<tr>
<td>Works only when the song is segmented into meaningful phrases, and people give meaningful query phrases</td>
<td>Works for any segmentation technique used</td>
</tr>
</tbody>
</table>

Thus, we have analyzed the performance of the Modified Dual-ternary indexing algorithm and the Multi-key hashing algorithm, for similarity, and exact matching with differing query lengths. It can be observed from the discussions given above, that the Multi-key hashing algorithm is more suitable for our system, where multiple components have been extracted from the input music, making it useful as keys for indexing. The results also showed the effectiveness of our features for representing music signals. The indexing scheme could be used for designing a web based music retrieval system with advanced search facilities, that could handle not only traditional text based queries or query by humming or query by example, but also queries specifying individual music and non music characteristics. Other types of data that could use the proposed indexing technique are discussed in Chapter 8.