

Real-Time Retrieval of Songs from Musical Databases with Query-by-Humming

Hsuan-Huei Shih, Tong Zhang, and C.-C. Jay Kuo
Integrated Media Systems Center and Dept. of Electrical Engineering – Systems
University of Southern California, Los Angeles, CA 90089-2564
hshih@usc.edu, {tzhang, cckuo}@sipi.usc.edu

1. Abstract

A real-time song retrieval system based on the humming of a person is investigated in this research. This system exploits the technique of pitch tracking to extract features of a humming, and then performs an enhanced string matching algorithm to filter out unrelated songs in an effective manner. A song database is constructed based on a hierarchical way to organize extracted features, which facilitates the song retrieval process. Experimental results show that the proposed feature extraction method, the string matching algorithm and the hierarchically organized database together can provide a very efficient solution.

2. Introduction

It occurs that we would like to find a song in a Karaoke room or a record store without knowing the name of the song. Sometimes, we may be able to get it successfully by browsing through a list consisting of names of composers and performers. This is however not always the case. It is therefore desirable to search a song by simply humming the tune of it. This technique is known as “query-by-humming”, which is a special topic in the research of content-based audio retrieval.

The best way for human being to find text based information from a database is to use keywords. For an image database, it would be helpful to use image features to retrieve desired images. There have been content-based image retrieval techniques developed recently, where a person retrieves relevant images based on a certain measure in terms of color, texture or shape similarity. The same idea can also be applied to the management of musical databases. Since the archival of digital music is growing very fast, how to use a natural way, such as query-by-humming, to retrieve a piece of music is becoming an important topic these days.

A query-by-humming system basically works as follows. The user hums a piece of query tune as the input to a

system. The system first translates the humming into a sequence of pitches. Then, string matching techniques are used to find the most similar songs in the database. Finally, retrieving results are returned to the user.

There are several important issues in the implementation of the query-by-humming system. First, how to track the humming pitches accurately is the most important problem for a query-by-humming system. If pitches are not properly tracked, retrieved results will not be correct in most cases. The second key issue is string matching. The tracked pitches will be recorded as a sequence of notes. This sequence is compared with songs in the database based on string matching techniques. Third, according to our experience, only the pitch information is not enough to retrieve the desired song from the database accurately. More side information of the hummed query is needed. More details will be discussed in later sections regarding the type of side information that will help string matching to achieve better accuracy. Finally, the music database should be indexed and organized based on features that are used in string matching.

3. System Overview

In this work, we propose a new approach of query-by-humming for musical applications. With this approach, the query can be processed in real time and the result has a rather high accurate rate. There are three major modules in the proposed system, i.e. the pitch tracking module, the string matching module, and the database module, as illustrated in Figure 1. For the input to the system, the user is asked to hum the first 12 to 15 notes of a song which is to be searched. Then, the hummed tune will be sampled and recorded at the sampling rate of 8 kHz, which is enough for the human voice. Next, some preprocessing will be done and the signal is down-sampled to 2 kHz since very few people can hum over 1 kHz. The system will record the hummed tune and perform pitch tracking on the fly. Afterwards, the tracked humming information is sent to the string matching module for comparison. Finally, a list of music compositions retrieved from the database module, which are ranked in the similarity order,

will be presented to the user. Details of each module are explained below.

4. Pitch Tracking Module

The first processing step in the query-by-humming system is to track the pitch of human humming. There are two major approaches to do pitch detection. One is in the time domain and the other in the frequency domain. Approaches in the time domain, such as the autocorrelation method (Equation 1 shown below), can be used to find the period of an input signal. The disadvantage of this method is that the complexity is too high. Also, it will cause a notable delay when pitches are being tracked. Among frequency domain approaches, the Fourier transform is the classical way to find the frequency of an input signal (see Equation 2). The music signal usually contains many frequency components including the fundamental frequency and its harmonics. In the proposed system, we consider a scheme called the Fundamental Period Measurement Method (FPM) [1] as shown in Figure 2, which is a frequency domain approach. The humming signals are passed through the FPM filter bank, and the estimated pitch values are obtained. This FPM filter bank has 40 subbands which are formed by 40 lowpass filters ordered from 1 to 40. The n^{th} FPM subband is equal to the n^{th} lowpass filter subtracted by the $(n-1)^{\text{th}}$ lowpass filter. These 40 lowpass filters are of the IIR Chebyshev II type. The reason to choose the IIR filter is that it may have a smaller order than that of the FIR filter to achieve the desired frequency response. It reduces the computational complexity.

Before the signal passes through the FPM filter bank, a pre-processing step is conducted. That is, the input humming signal is first segmented into notes. This segmentation is essential in getting a good pitch detection result and may provide some useful side information. Two approaches have been tested, i.e. fixed segmentation and dynamic segmentation. In fixed segmentation, we choose 20ms to 100ms as the segment period, and pass the segmented signal into the FPM system. Sometimes such a segmented period may contain more than two major frequency components during the transition period of the input signal. FPM pitch detection will not work properly in such a case. The tempo information of the original input humming will be destroyed and lead to more problems in the string matching module. Another disadvantage of fixed segmentation is that the complexity will increase when the segment period decreases. Thus, we may consider dynamic segmentation instead, where a period of hummed note is used to segment the input humming. In other words, the input humming will be segmented by its notes. In this way, the original tempo information of the input humming will be kept. We use dynamic segmentation in

our system and the segmentation is done based on the amplitude of the input signal. The user is asked to sing or hum a particular sound of 'da' or 'ta' when needed [2],[7]. The reason that these two sounds are selected is that they will cause a drop in the amplitude of a 60ms duration or longer, and this drop of the amplitude will help the system to segment the input signal. Each obtained signal segment is represented as one note of the humming. The amplitude threshold for deciding a segment is dynamically changed with the input signal. As the output of the pitch tracking module, the duration (interval contour) and the pitch detection result of each note (melody contour) are recorded.

The pitch of each segment is estimated based on the energy distribution of 40 subbands. The energy of each subband is calculated after the segmented signal passes through the filter bank. The energy distribution of the 40 subbands of a segmented signal may have several peaks, as shown in Figure 3. These peaks appear periodically, indicating the energy of the fundamental frequency and those of harmonic frequencies. The subband is selected in which the peak of the fundamental frequency falls, and the pitch will be calculated according to the output of the selected subband.

Human beings recognize a song mostly based on the pitch relation between neighboring notes and the duration of each note. For example, when a song is played in two different keys, people normally think that they are the same. For two music compositions that have the same combination of pitches but the duration lengths of corresponding pitches are different, people will recognize them as two different songs. Also, the tempo of humming a musical piece may vary a lot for different people, but the duration relationship between two adjacent notes do not change too much. Although fine details of pitch and tempo information of each note will give the absolutely correct result [5],[6], no one can hum a piece of music exactly the same as the original music in the database. Therefore, according to the Mother Nature of the human brain, a coarse melody contour and an interval contour are adopted in the system. Also, it is observed that the length of melody contour of about 12 to 15 notes is sufficient for retrieval [2],[3],[7].

To obtain the coarse melody contour[3],[12], only the relationship between neighboring notes is recorded in the system. Three symbols are used to represent the relationship, i.e. U (the pitch of the current note is higher than that of the previous note), R (the pitch remains the same as that of the previous one), and D (the pitch is lower than that of the previous one). Let us see an example. The melody contour of the introductory theme of Beethoven's Fifth Symphony is represented as “* R R D U R R D” [3],

where the first symbol * denotes the beginning note of a song for which there is no reference available.

In the interval contour, the duration difference between each note and its previous note is recorded. A similar representation to that of the melody contour is used here. There are also three symbols defined: S (the duration of the current note is shorter than that of the previous note), T (the duration is the same as that of the previous one) and L (the duration is longer than that of the previous one). For example, the interval information of the introductory theme of Beethoven's Fifth Symphony is represented as "* , T, T, L, S, T, T, L".

5. String Matching Module

The second module in the system is the string matching module. A number of papers have been published on string matching techniques [4],[6]-[11]. There are two major features the proposed approach. One is histogram elimination while the other is approximate string matching.

In histogram elimination, the histogram information of the melody and interval contours is used to filter out most music pieces in the database which are unlikely to match the query before going to the string matching step. Four types of histogram information are used. They are the histogram of melody contour symbols (U, R, and D), the histogram of interval contour symbols (S, T and L), the run length histogram of melody contour symbols, and the run length histogram of interval contour symbols. Let us take the following two songs, "Two Tigers" and "Little World", for example. In "Two Tigers", there is "do-re-mi-do-do-re-mi-do-...", and in "Little World", there is "do-re-mi-do-mi-do-mi-re-...". Their histograms are plotted in the Figure 4. Although these two songs have very similar representations in notes, the system can distinguish them at the first stage through histogram analyses. A cost value is obtained regarding the similarity of the string from the database with the target string from humming. Higher cost indicates that the string is more unlikely to match the target string, while lower cost represents higher similarity to the target string.

Approximate string matching is used because it is difficult for most people to hum a piece of music accurately and errors may occur in the pitch tracking process. Exact string matching does not work well here. There are four major types of error that may occur when people hum and when the system performs pitch detection. They are: one or more notes accidentally added, one or more notes accidentally removed, several notes of the same pitch accidentally replaced by one note, and one note

accidentally replaced by several notes of the same pitch. All of these errors will cause mismatch of the strings. Therefore, we consider five operators for string matching. They can be classified into two categories: basic operators and advanced operators [2],[7].

The basic operators include Deletion, Insertion, and Substitution. The Deletion operator can be used to correct the error caused by accidentally adding a note. The Insertion operator can be used to correct the error caused by accidentally removing a note. The Substitution operator is to replace a note with another note, and it can be used to correct the error caused by incorrectly measured note.

The advanced operators for the musical sequence include Consolidation and Fragmentation. The Consolidation operator is used when the error is caused by user's humming more notes with the same pitch than it should be, or the pitch tracking module accidentally adds some segmentation points in the humming signal. The Consolidation operator corrects such an error by combing these notes together. The Fragmentation operator is used when several notes with the same pitch are merged together into one note. It usually happens when users hum one note instead several notes, or when users hum the segment tone not very clearly so that the pitch tracking system is not able to find the segmentation.

Each operator has its cost. The cost of an advanced operator is less than that of a basic operator, because the impact of adding, deleting, and replacing a note will cause a notable difference in the perception of people. Therefore, more cost indicates that the change causes a larger difference. The impact of advanced operators is less than those of basic operators and sometimes people cannot even tell the difference when a note is divided into several notes or when several notes are merged into one. The cost may be fixed or variable depending on the symbols involved. These operators are used to match the melody contour and the interval contour information of music pieces to the target pattern. Then, costs of all operators are added together. The smaller the cost is, the higher the chance it matches.

Histogram elimination can help the system to decide which strings from the database have a higher priority to be performed at the second stage, i.e. the approximate string matching, and postpone the most unlikely ones to the last. The system uses the weighted sum of costs from the two stages to obtain the most likely strings from the database. Finally, the system will generate a list of music pieces (represented by pre-required numbers) which are most likely to match the input humming from the user.

6. Database Module

The last module is the database module. We have used the MIDI format in building our database. There are several reasons for choosing the MIDI format instead of WAV files. First, in the comparison of the information recorded by both formats, MIDI files record the information of different instruments, i.e. what to play and when to play, while WAV files only record the waveform of the music. In other words, the pitch and interval information is contained in the MIDI files explicitly but not in the WAV files. Second, the size of the MIDI file is much smaller than that of the WAV file with the same music. It can therefore reduce the size of the database. It is much easier to do pitch tracking with MIDI files than with files in the WAV format. We first select the channel which contains the main melody and then get the pitch information from its data bitstream. For files in the WAV format, more processing is needed to transform WAV files into the corresponding pitch and tempo information.

A hierarchical musical database is built according to the histogram information as shown in Figure 5. There are MIDI files of more than two hundred songs and one hundred pieces of musical instrument performance in our database at the current stage. We continue to gather more MIDI files to enlarge our database now.

7. Experimental Results

Two male's and Two female's vocal was tested in the system, and 200 General MIDI music sources from public domain sources is stored in our database. The experiment was focused on the pitch tracking module and the string matching module. Experimental results showed that the pitch tracking module could achieve an accuracy rate as high as 99% under an ideal condition, where users hum the segmentation tone 'da' or 'ta' clearly. The accuracy rate of men's voice is higher than the accuracy rate of women's. The first note of the humming is not easy to be hummed accurate for people. Pre-humming a tone before the first note will help to correct this problem. For the stream matching module, it will do the histogram comparison and approximate string matching and the goal is that it will return a list of ranked music according to the probability of matching. A user is able to use the melody contour and the interval contour to retrieve the music from the database. The total cost of retrieving music will then convert to scores from 0 to 100, which 0 means the lowest possibility and 100 means the highest possibility among the returned list. The user can pick the returned music title according to the score of each music title. To perform the retrieval without the first feature, i.e. histogram elimination, would cost more time in returning the

retrieval list of music title. However, if both first feature and second feature are used, some highly matched music may be postponed to a later stage. It is a tradeoff between choosing a faster response and a higher accuracy rate.

8. Conclusion

In this work, we proposed a system that combines pitch detection, string matching and a hierarchical database structure. The system improves the accuracy of pitch detection of human vocal and the performance of string matching as shown in experimental results. In the future research, we will focus on improving the accuracy of pitch detection of both male's and female's vocal under more real world conditions. We would also like to see if the MP3 file format has some features can be used in the proposed Query-by-Humming system.

9. References:

- [1] William B. Kuhn, "A real-time pitch recognition algorithm for music applications", *Computer Music Journal*, Vol. 14, No3, Fall 1990.
- [2] Rodger J. McNab, Lloyd A. Smith, Ian H. Witten, "Signal processing for melody transcription", *Proceedings of the 19th Australian Computer Science Conference*, Melbourne, Australia, Jan. 31- Feb. 2 1996.
- [3] Asif Ghias, J. Logan, D. Chamberlin, and B. C. Smith. "Query by humming: musical information retrieval in an audio database", *Proceedings of ACM Multimedia Conference'95*, San Francisco, California, November 1995.
- [4] Jia-Lien, Arbee L. P. Chen, and C. C. Liu, "An Approximate String Matching Algorithm for Content-Based Music Data Retrieval", *Proceedings of IEEE ICMCS99*, 1999
- [5] Hsu J.L., and C. C. Liu, "Efficient Repeating Pattern Finding in Music Database", *Proceeding of Seventh International Conference on Information and Knowledge Management (CIKM'98)*, 1998.
- [6] Arbee L. P. Chen, and C. C. Liu, "Music Databases: Indexing Techniques and Implementation", *Proceedings IEEE Intl. Workshop on Multimedia Data Base Management Systems*, 1999.
- [7] McNab R.J., Smith L.A., Witten I.H., Henderson C.L. and Cunningham S.J., "Toward the Digital Music Library: Tune Retrieval from Acoustic Input", *Proceeding Digital Libraries '96*, pp 11-18, 1996.
- [8] Bainbridge D. and Inglis S., "Melody based tune retrieval over the World Wide Web", *Working Paper 98/17*, Department of Computer Science, University of Waikato; November, 1998.
- [9] Ricardo Baesa-Yates and G.H. Gonnet, "Fast string matching with mismatches", *Information and Computation*, 1992.

[10] Ricardo Baeza-Yates and Gonzalo Navarro. "A Faster Algorithm for Approximate String Matching", In D. Hirschberg and Gene M. (editors), *Proceedings of CPM'96*, LNCS 1075. Pages 1-23, 1996.

[11] Alexandra L. Uitdenbogerd, Justin Zobel, "Manipulation of Music For Melody Matching", *Electronic Proceedings ACM Multimedia 98*, 1998.

[12] Mongeau, M. and Sankoff, D. "Comparison of musical sequences," *Computer and the Humanities* 24:161-175, 1990.

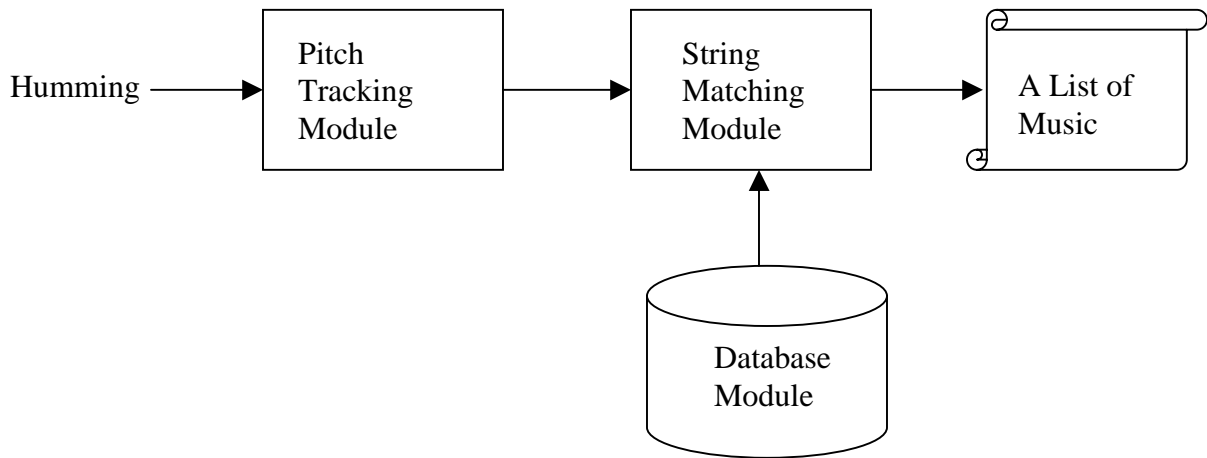


Figure 1: Framework of the proposed query-by-humming system.

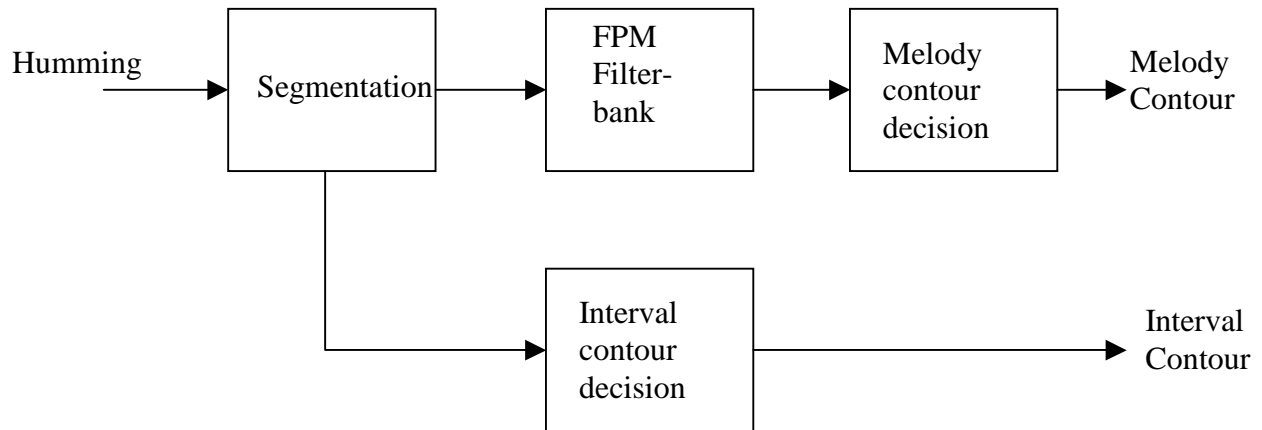


Figure 2: Structure of the pitch tracking module.

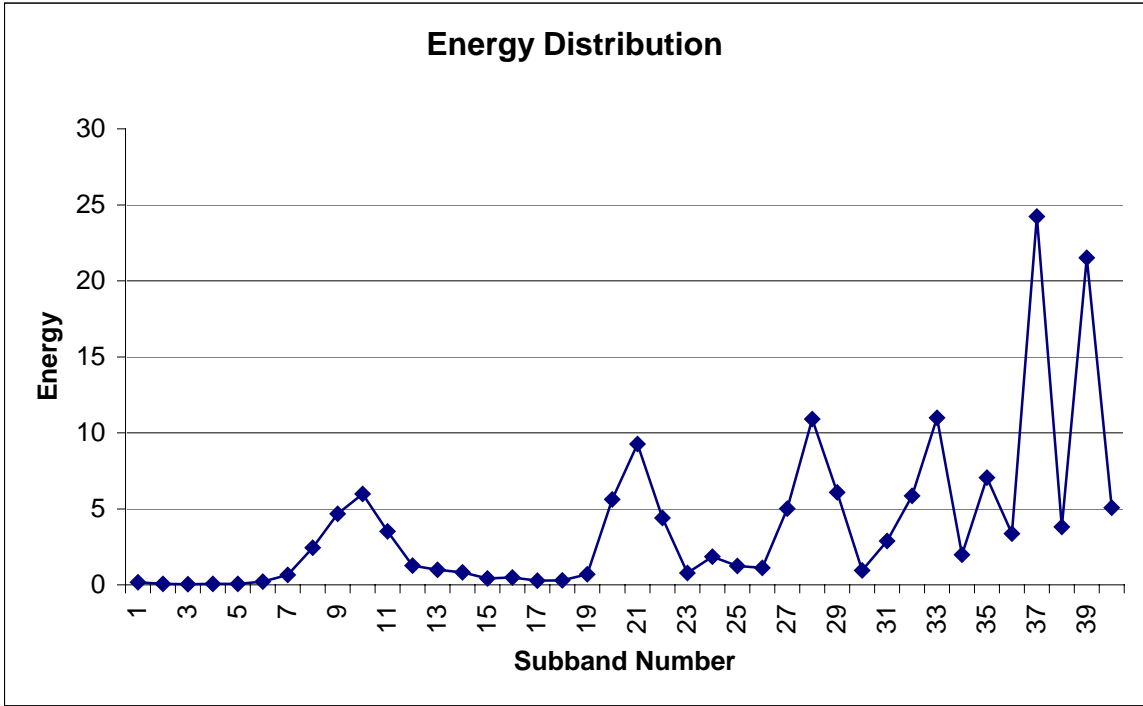


Figure 3: The energy distribution of 40 subbands.

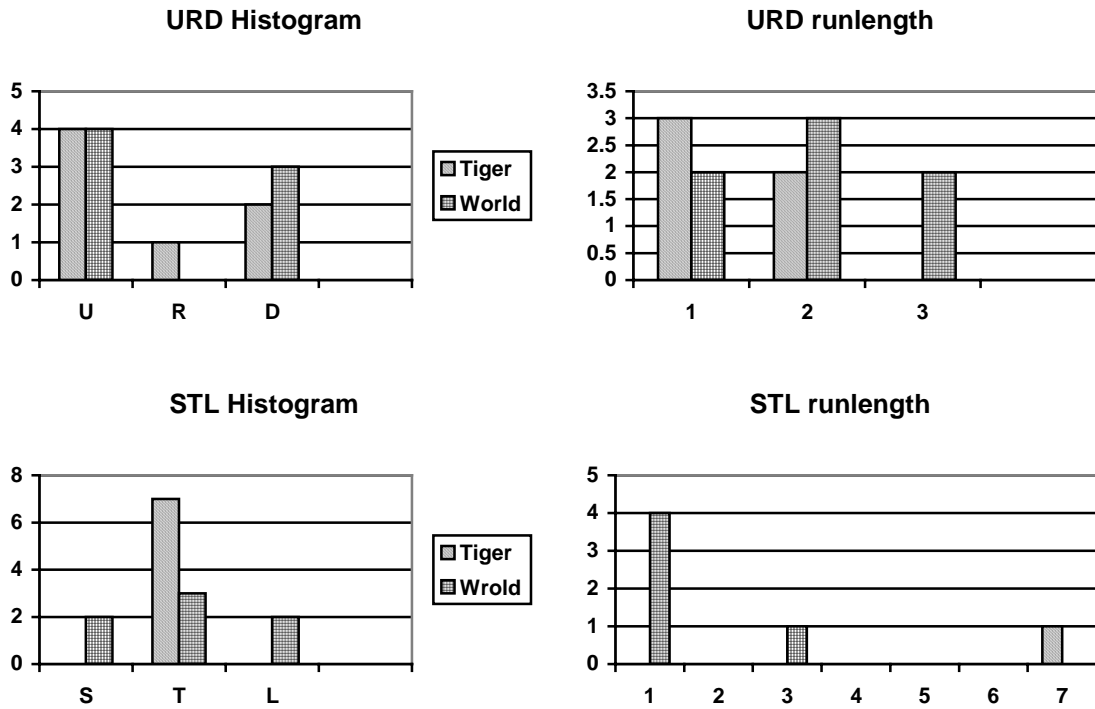


Figure 4: Histograms of "Two Tigers" and "Little World".

Two tigers (T):
12311231
*UUDRUUD

Beethoven's
5th Symphony (B):
5552 # 4442
*RRDURRD

Happy birthday (H):
11214311
*RUDUDDR

ROC Nation Anthem
(R):
11335532
*RURURDD

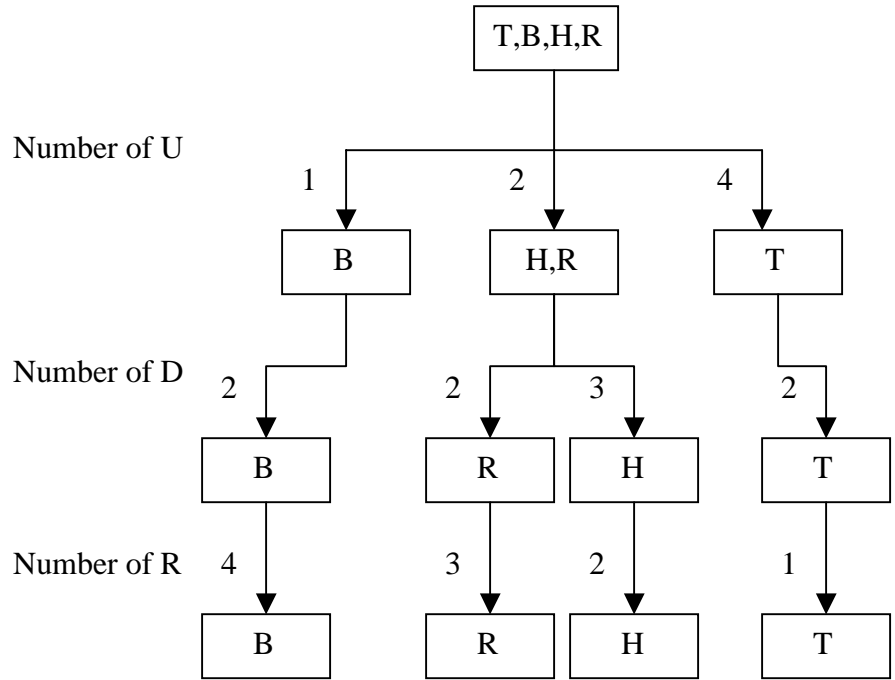


Figure 5: Hierarchical database according to melody contour (U, D, and R) histogram of 8 nodes.

$$R_n(k) = \sum_{m=-\infty}^{\infty} x(n+m)w_1(m)x(n+m+k)w_2(m+k)$$

Where

$$w_1(m) = \begin{cases} 1 & 0 \leq m \leq N-1 \\ 0 & \text{otherwise} \end{cases}$$

$$w_2(m) = \begin{cases} 1 & 0 \leq m \leq N-1+k \\ 0 & \text{otherwise} \end{cases}$$

Equation 1: Windowed Autocorrelation

$$F(k) = \sum_{n=0}^{N-1} f(n)e^{-j2\pi nk/N}$$

Equation 2: Discrete Fourier Transform