This is the published version


Available from Deakin Research Online

http://hdl.handle.net/10536/DRO/DU:30009585

Reproduced with the kind permission of the copyright owner

Copyright: 2003, International Society for Computers and Their Applications
Abstract

With phenomenal increases in the generation and storage of digital audio data in several applications, there is a growing need for organizing audio data in databases and providing users with fast access to desired data. This paper presents a scheme for the content-based query and retrieval of audio data stored in MIDI format. This is based on extraction of melody from the MIDI files and suitably comparing with the melody of the query. The results of retrieval using the proposed algorithm are presented.

1. Introduction

The explosive increases in the amounts of audio data being generated, processed, stored, and used in several computer applications have created a critical need for the fast and accurate retrieval of the desired data. Predominantly, there are two kinds of audio data generated and consumed by human beings - speech and music. These have distinctly different characteristics which could be suitably used in the indexing, search, and retrieval process. In this paper, we concern ourselves with music data stored as MIDI (Musical Instrument Digital Interface) format files. Example applications using huge amounts of music data that need to provide content-based query and retrieval facilities include digital libraries, entertainment industry, virtual reality, etc.

In this paper, we propose a scheme for the content-based query and retrieval of music data stored as MIDI format files. The proposed scheme is based on the extraction of 'melody' from a MIDI data file and using a suitable scheme for determining a match of melodies in the data to those in the query. The next section briefly describes the MIDI format and a few schemes for content-based retrieval of music data. Section 3 presents the proposed algorithm. Experimental results are given in Section 4, followed by conclusions.

2. Background and Related Systems

This section presents a brief introduction to MIDI and describes a few schemes for the content-based retrieval of music data.

2.1 MIDI Format

MIDI stands for Musical Instrument Digital Interface. One view of MIDI is that it essentially represents a language for the generation of sound that constitutes a given piece of music. The various components of the sounds such as the pitch, duration, different combinations, etc., are clearly specified. Thus, any instrument with a MIDI interface can appropriately interpret the MIDI description and generate the sounds which results in the desired music being played. Thus MIDI format (also called encoding) does not 'store' the music data, but clear instructions about how the music should be produced. The MIDI encoding includes the specification for the musical instrument, the beginning and end of a note, basic frequency, and sound volume. MIDI format allows for 128 notes. The MIDI data is grouped into MIDI messages. Each MIDI message communicates one musical event. The event are basically actions such as pressing keys, moving slider controls, adjusting foot pedals, etc. The MIDI standard identifies 128 instruments. A MIDI device (musical instrument) is mapped to a channel. The number of simultaneously played notes per channel can range from 3 to 16. MIDI format also enables the editing of the music easily.

2.2 Content-Based Retrieval of Music Data

A scheme for the classification of audio data based on the content, to facilitate subsequent search and retrieval, is presented in [8]. In this scheme, the sound is first analyzed and reduced to a small set of acoustical attributes to which statistical techniques are applied to do classification and retrieval. The acoustical attributes of sound that are analyzed are: pitch, loudness, brightness, bandwidth, and harmonicity. These could be directly derived from measurable attributes of sound. Then features such as average, variance, and autocorrelation are computed (and usually weighted by the amplitude trajectory) and stored. In addition, the duration of the sound is also stored. The feature vector thus consists of average, variance, autocorrelation, and duration. This is referred to as N-vector. The components of the feature vectors are all numbers and are the only information used in the content-based classification and retrieval of sounds. The acoustical features extracted and used for search are short or single-gestalt sounds. Also, simultaneously sounding sources are treated as a single ensemble.
In [2], a system for search and retrievals for query-by-example is described. This supports only music data and the query is a humming of some tune. Their approach is based on the observation that melodic contour, which is a sequence of relative differences in pitch between successive notes, can be used to discriminate between melodies. The relation between successive pitch changes is denoted by a symbol in the alphabet \((U, D, S)\), signifying that the (current) pitch is above, below, or same as the previous pitch, respectively. The data analyzed for these pitch changes and a sequence of symbols based on the above relation is derived. This forms a string of symbols. The query is similarity analyzed and a pattern of symbols is derived. Then well known approximate pattern matching algorithms are used to find the set of strings which closely match the given pattern. The disadvantage, however, is that the pitch extraction is computationally intensive and is not very accurate.

A scheme for the classification of audio data and content-based retrieval based on the classes was presented in [7]. This scheme was based on a few time-domain and frequency-domain characteristics of the audio data. Short-time DCT applied to blocks of audio data yield transform coefficients which have several desirable properties including energy compaction and noise localization. The appropriate subsets of coefficients are used as features representing the predominant characteristics of the data that can be used as indices in content-based retrieval.

A system for query by humming for a musical database is described in [5]. The voice input is first filtered for noise reduction. A processing module then tracks the pitches and assigns note numbers which are similar to MIDI note numbers. These are then compared to pre-processed input data using similarity measures. A scheme for extracting melody from MIDI files is described in [6]. It uses a modified Lempel-Ziv algorithm to extract repetitive patterns in the music data to derive melody. (Lempel-Ziv algorithm is a dictionary-based technique used in data compression). Another scheme for content-based retrieval where sounds are characterized by templates derived from a tree-based vector quantizer is described in [1].

3. The Proposed Audio Classification Scheme

The overall architecture of the proposed system is shown in Fig. 1. It consists of an index of melodies. This would have been done by an offline process. The query is input from a keyboard, and is subjected to processing to derive melodies. The search and retrieval subsystem performs an approximate matching of the melodies in the query against the melodies in the data and retrieves the closest matching data items. Thus, the melodies serve as indices which form the basis for comparison. The key elements of the system are the process of extracting the melodies from the audio data in MIDI format, and the comparison of melodies for similar-sounding parts of the audio data. In this paper, melody is roughly defined as the tune which can serve as an identifying feature of a given audio clip.

![Figure 1: Architecture of the proposed system.](image)

3.1 Melody extraction from MIDI

The melody extraction process should derive the predominant tune which can form a distinctive feature of the underlying audio. Due to a lack of clear cut definition and computational schemes for melody, the process of melody extraction is not clearly laid out. In the proposed scheme, a set of tracks that capture the main melody are identified. In general, tracks containing percussion, chords, and non-melodic information are identified and discarded. Chords are lower octave keys that are played with the main melody. They could be on a separate track or on the same track as the main melody. In the former case, their removal is easier. In the latter case, they correspond to lower octave notes and have to be identified and removed. Also, tracks containing less than about 10 events are treated as not contributing to the melody and are discarded. The melody extraction is then performed on the selected tracks. The melody is defined to correspond to melodic contour which is essentially a string of symbols from the alphabet \(\{S, U, D\}\). The differences in successive pitches are derived and denoted by \(S, U, D\), depending upon whether (a) they are very low, corresponding to successive pitches being similar, (b) significant positive number, corresponding to a pitch higher than the previous one, and (c) significant negative number, corresponding to a pitch lower than the previous one. This aspect is similar to the scheme proposed in [2]. The melody information above is added to a database. This information is used in the search and retrieval process. An overview of the process is shown in Fig. 2.

![Figure 2: Melody extraction from MIDI files.](image)
3.2 Melody comparisons

Recall that the melody information in a given audio clip in MIDI format is ultimately extracted and represented as a string of symbols from the alphabet \( \{S, U, D\} \). A similar process is used to extract melodic information from a given query is thus reduced to a problem of performing pattern matching of the query string with the data strings. It should be noted that the matching should accommodate similarities and not just support exact matches. This is inherent in any matching scheme for subjective and expression-rich data such as audio, image, and video. A scheme similar to "agrep", an approximate string matching algorithm is used. The search and retrieval process also keeps track of the amount of difference, and collects all data above a certain threshold of similarity. It also keeps track of the position(s) in the data where a match for the query occurs. The results are ranked in decreasing order of similarity. A user interface provides a means of playing the data portions containing a match to the given query.

4. Experimental Results

The proposed scheme has been implemented in MATLAB and tested on a collection of nearly 150 audio files on a 733 MHz PC. The files were of varying durations. The music files consisted of several varieties ranging from single instruments to ensembles. On the average, the search process took about 2.3 seconds for the search process. Note that this could be speeded up by organizing the indices in suitable data structures and by implementing the search algorithm in C or C++.

The major objective of the proposed study is the feasibility of effectively using data in MIDI format for content-based retrieval of audio data. The search time as function of the database size is shown in Fig. 3.

![Figure 3: Search Times vs. Database Size.](image)

4.1 Accuracy of Results

The results of this system are very promising for the area of content-based audio retrieval. The results show that the system is capable of coming up with correct matches. There are two metrics used for content-based retrieval systems. The 'Precision' metric and the 'Recall' metric is an indicator for the effectiveness of the system.

Table 1 presents the set of matches for two different queries.

<table>
<thead>
<tr>
<th>Category of Query</th>
<th>Name of Song in Query</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Children's Songs</td>
<td>Old MacDonald</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Yankee Doodle</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>O Susanna</td>
<td>0.667</td>
<td>1</td>
</tr>
<tr>
<td>National Anthems</td>
<td>India</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>Algeria</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>United States</td>
<td>1</td>
<td>0.5</td>
</tr>
</tbody>
</table>

The query results for O Susanna included two other songs with the title 'Farmer Dell'. Upon further investigation, it was found that the 'Farmer Dell' songs had title pieces that were similar to the starting of 'O Susanna'. Figure 5.3 shows how the search times vary with increasing number of files. Five groups of 100 files each, with different average playing times of each group were considered. It can be seen that search times increase with increasing playing times.

4.2 Effect of Query Size on Search Quality

As in the case of search time, the quality of retrievals is also affected by doubling the query size. The number of false alarms went down, but the number of misses increased.

![Figure 4: Precision & Recall vs. Query Size](image)
4.3 Effect of Query Size on Search Time.

The size of the query has a significant effect on the search times. Figure 5 shows the effect of query size on search times. The normal query size used for evaluating precision was 12 events. On doubling the size, the search times could be reduced by a great extent. The average retrieval time for a query size of 12 was 5.012 seconds. On doubling the query size to 24, the average retrieval time dropped to 1.024 seconds.

![Figure 5: Search Times vs. Query Size](image)

4.4 Discussion of Results

This system has been implemented using Matlab 6.0 on a Pentium 733 MHz Computer. Note that Matlab is a scripting language and is inherently slow with processing data. If the system is implemented using a high level language, search times will be significantly lower.

The overall search times keep increasing with an increase in the number of melody files in the database. This is to be expected as the matching algorithm has to go through more strings and make more comparisons. The search times are proportional to the playing times of the MIDI files in the database. Figure 3 shows five series with each representing an average playing time for 100 MIDI files. With an increase in the playing times of the files, the search times also increase.

The effect of query size on the quality of results shows that the quality gets affected with the increase in query size. That is, for a bigger query size, the system tends to 'miss' more often than for a smaller one. This can be explained by the fact that a bigger query size naturally translates into a more accurate representation of the input. This leads to fewer matches.

The other effect of query size is on the search times, which seem to come down with an increase in the size of the query. This is an inherent quality of 'agrep', the approximate string matching algorithm used in the system. The algorithm shifts from position to position looking for matches. With bigger query sizes, there are more chances of making bigger shifts from these positions which means that the agrep algorithm looks through the melody files faster. This translates into lower search times.

5. Conclusions and Future Directions

Fast and accurate content-based retrievals of audio data are important in several applications. This paper presented a scheme for the content-based retrieval of audio data consisting of instrumental music stored in MIDI format. The scheme identified and filtered out the tracks which did not contribute to melody of the underlying data. It then derived melodic contour from the significant tracks, which was used as the search feature. The retrieval precision was fairly good.

Inclusion of tempo in the audio data for matching would be one of the future directions. This could be used as first level filter. Building more robustness in the detection of predominant tracks, speeding up the search process, and effectively using the repetitive nature of music data in the search process are a few of the other future directions.

References