

# Content-based Retrieval in MIDI and Audio

Michael Clausen, Frank Kurth, Roland Engelbrecht

Institute of Computer Science III, University of Bonn

---

## Abstract

*This paper briefly reports on recent advances in the development of search engines for music and audio within the MiDiLiB project: while notify! is a tool for searching in symbolic music data like MIDI, audentify! allows to search in PCM audio.*

---

## 1. Introduction

Content-based search in musical data is still a great challenge. This is mostly due to the fact that the cognitive foundations of music are still not well understood. Therefore, almost all music database system supporting content-based search have to rely on musical data which is meant to describe music for reproduction. This includes scores for human performances and technical formats for reproduction by machines, such as PCM or MIDI. Even scores do not represent cognitive elements explicitly.

Hence, almost all search-engines based on score-like data depend on the syntactical representation used and try to search by syntactical means. Often some musical objects such as melodic lines are extracted manually, as it is done in MELDEX<sup>1</sup>, Themefinder<sup>3</sup>, SEMEX<sup>5</sup> and ECHO<sup>7</sup>. Within our MiDiLiB-project<sup>8</sup>, some efforts have been made to circumvent the extraction of musical objects as such. In contrast to other approaches, our notify!-search engine, allows polyphonic search, i.e., both, query and database pieces can be polyphonic. This has been achieved by taking a novel point of view. Whereas the above mentioned projects take essentially a string-based approach, we prefer a set-theoretical viewpoint. The OMRAS project<sup>6</sup> seems to pursue a comparable strategy.

Though notify! had been primarily developed for polyphonic music retrieval, it can be easily applied to monophonic data, too. In order to obtain monophonic data from polyphonic data, we are currently developing a tool named melodify! which supports semi-automatic melody extraction.

Since for a score-based search engine you must own some kind of score-related data, many applications simply cannot

work if only waveform-related data are available. Whereas score-related music data offers some minor kind of musical objects, i.e., notes and the like, waveform-related representations even lack those simple objects, at least explicitly. As an algorithmic transcription from waveform to score-like representations in the general case is a very hard task and yet not fully understood, one usually performs some kind of feature extraction on the waveform data, using the features for indexing and searching purposes.

This path is also pursued by our audentify! tool. In contrast to other approaches, it operates data-driven, i.e., the size of the feature list per time-unit depends on the data.

In the sequel we briefly describe both search engines and present some performance figures.

## 2. Searching in MIDI

The most important items of a score are notes and rests. These are the objects which are mostly searched for in music databases, while dynamics and agogics are usually ignored. The major property of notes is their constellation, thus their vertical and horizontal relationships to each other. This can be easily visualized in the usual piano roll representation of a score, which can be considered as a cartesian coordinate system with time along the x-axis and pitch along the y-axis. Then a note is simply a line segment parallel to the x-axis. The left edge of the segment denotes the onset time whereas its width is proportional to the note's duration. The segment's vertical position indicates the pitch.

In other words, a score can be represented as a set of horizontal line segments (ignoring all articulatoric marks, of course). The same is true for any excerpt of a score: it is

a	4	6	8	10	12	14	16	18	20	30	50	100
b	51	70	86	87	92	93	97	96	100	107	125	159
c	1	3	5	6	7	8	10	11	12	19	31	64

**Table 1:** The average total system response time (row b) in milliseconds depending on the number of notes in the query (row a). Row c describes the time required for disk access to fetch inverted lists. Most of the remaining time (total response time – disk access time) is consumed by decompressing inverted files (Pentium II, 333 MHz, 256 MB RAM, Windows NT 4.0).

also just a set of horizontal line segments. This view is completely different from the approaches used in other music database systems so far, where string-based representations prevail.

Exploiting the chosen representation, a content-based query  $Q$  to the database is also a set of line segments and therefore the music search can be reduced to a search for supersets, i.e., all music pieces  $P$  in the database with  $P \supseteq Q$ . At least, this is the general idea. On one hand, the searched fragment can start anywhere in a database, not just at the beginning, and it may be transposed. Thus, actually, we are looking for all  $P$  with  $P \supseteq Q + x$ , where  $x$  is a suitable shift vector in time and pitch.

On the other hand, one has to take into account errors in both, the query and the database pieces, for example wrong timing and pitches as well as missing and superfluous notes. More technically speaking, one has to allow fuzzy query notes and one has to offer some kind of mismatch search. All of this is accomplished by our software `notify!` which will be described here briefly, more details can be found in <sup>2</sup>.

For the sake of simplicity, `notify!` only regards the onset times of the notes but in addition the metrical position of each note is evaluated. Temporal shifting is only allowed measure-wise. The pair of metrical position and pitch is used to index each note in a so-called inverted-file which is originally known from full-text retrieval systems. For each existing pair  $(M, p)$  of metrical position  $M$  and pitch  $p$  there is a set  $L(M, p)$  which refers to all notes in the database with these characteristics. Actually, all database pieces are logically concatenated such that we only have a very large piece of music. Therefore, it suffices to store only the measure number of the according notes in  $L(M, p)$ . The actual database piece can be easily reconstructed from the “global” measure number and the number of the starting measures of the pieces within the global piece. Hence,  $L(M, p)$  is simply a set of measure numbers, which can be compressed efficiently: `notify!` spends only 0.7 MB of memory for one million notes.

A query  $Q$  is decomposed in a similar manner: we obtain a sequence  $(M_i, m_i, p_i)$  representing the query notes with  $M_i$  being the measure number,  $m_i$  the metrical position and  $p_i$  the pitch of the  $i$ -th note. In the most simple case, we obtain the set of matches  $M(Q)$  as the intersection of the involved lists.

This can be extended to support fuzzy notes and  $k$ -mismatch search. A *fuzzy note* is a set of notes where each element represents a possible position for the “note event” in question. Of course, each element of the fuzzy note can be weighted according to its importance. In *k-mismatch search*, up to  $k$  notes may be missing in the database piece.

Finally, we address some performance issues of the `notify!` system. For our test database, consisting of 33 million notes extracted from over 12,000 MIDI files, the index consumes only 22 MB of disc space. Typical response times are listed in Table 1. The overall time to create this index was as small as 40 seconds. This offers the possibility of creating an index which meets the personal demands of a certain user in an acceptable amount of time.

### 3. Searching in Audio

For PCM data we consider the task of locating all occurrences of a fragment of music or some other kind of audio signal in a large collection of audio signals. The term “occurrence” has to be understood as a near-exact match, thus allowing some distortions, e.g. introduced by analog transmission or lossy compression.

Our software `audentify!` <sup>4</sup> performs this task by extracting a list of feature values from each audio signal. Whereas most of the other approaches use a fixed number of features for each audio frame, `audentify!` operates data driven, i.e., the length of the feature list depends on the “local highlights” of the signal in question.

The progression of the features is analyzed and each feature is assigned a feature class. We obtain a situation similar to that in `notify!`: we can construct an index list  $L(c)$  listing the locations of all features belonging to class  $c$ . Consequently, one can implement the very same techniques for `audentify!` as in `notify!`. The major difference is the notion of fault-tolerance.

`audentify!` has been tested with a collection of 50 GB of audio data from 930 songs. The index was as small as 58 MB, this is only 1/862 of the original data. This can be reduced to down to 1/3500 when only one stereo channel is indexed and a suitable feature setting is used while retrieval results still remain convincing. Processing a query of lengths between 0.5 and 3 seconds require only 0.5–1.5 seconds. For further information, we refer to our paper<sup>4</sup>.

#### 4. Conclusion and Future Work

We have developed a unified framework for content-based search in multimedia documents. The search engines `notify!` and `authenticate!` are special cases of this framework, however in each case one has to solve additional application-specific problems. In a similar way we are currently developing search engines for images and video.

#### References

1. David Bainbridge. *MELDEX: A Web-based Melodic Index Service*. In: *Melodic Similarity: Concepts, Procedures, and Applications*. *Computing in Musicology*, volume 11, chapter 12, pages 223–230. MIT Press, 1998.
2. Michael Clausen, Roland Engelbrecht, Dirk Meyer, and Jürgen Schmitz. PROMS: A Web-based Tool for Searching in Polyphonic Music. In *Proceedings Int. Symp. on Music Information Retrieval 2000, Plymouth, M.A., USA, 2000*.
3. Andreas Kornstädt. *Themefinder: A Web-based Melodic Search Tool*. In: *Melodic Similarity: Concepts, Procedures, and Applications*. *Computing in Musicology*, volume 11, chapter 13, pages 231–236. MIT Press, 1998.
4. Frank Kurth and Michael Clausen. Full-text indexing of very large audio data bases. In *Proc. 110th AES Convention, Amsterdam, NL, 2001*.
5. Kjell Lemström. *String Matching Techniques for Music Retrieval*. PhD thesis, University of Helsinki, 2000. Department of Computer Science, Series of Publications A, Report A-2000-4.
6. OMRAS web site. [www.omras.org](http://www.omras.org).
7. Tomonari Sonoda and Yoichi Muraoka. A www-based melody-retrieval-system — an indexing method for a large melody database. In *Proc. ICMC 2000, Berlin*, pages 170–173, 2000.
8. MiDiLiB. Content-based Indexing, Retrieval, and Compression of Data in Digital Music Libraries. <http://leon.cs.uni-bonn.de/forschungprojekte/midilib/english/>.