Audio Engineering Society
Convention Paper
Presented at the 110th Convention
2001 May 12–15 Amsterdam, The Netherlands

Full-Text Indexing of Very Large Audio Data Bases

Frank Kurth and Michael Clausen
University of Bonn, Department of Computer Science
Bonn, 53117, Germany

ABSTRACT

We present a system for indexing of and index-based search in PCM-based audio material. Given a short excerpt of a waveform signal as a query, the index returns all pieces in a data base containing that waveform. Additionally, the precise position of the waveform within those pieces is returned. The indexing method is robust against several signal processing operations such as lossy compression or analog transmissions. Indexing of a test data base consisting of approximately 50 GB of audio data results in an index of size 58 MB. Response times to queries of lengths of about one or a half of a second are only fractions of a second.

INTRODUCTION

Today, enormous amounts of digital audio data are stored in digital libraries and archives. The purposes of storing the audio data are manifold including preservation, documentation, backup, or further processing. Moreover, the amount of audio data to be handled by audio data base systems is significantly increased every day. For an example we may think of radio- or TV stations archiving their daily programmes.

Unfortunately, in comparison with conventional text-based data base systems, todays audio data base systems lack suitable indexing and search functionalities. Therefore, although stored in a data base, much of todays audio material is inaccessible to someone searching for it. What makes things more complicated is that 'audio queries' are not as easily specified in a textual form as queries to a conventional data base. In fact there are many different possibilities of how an audio query could look like and there are also many different expectations behind those queries. For example, a person looking for a popular song while only memorizing the song's main tune would probably accept several versions of that song including its original version, a live recording, or even a cover version by a different performer. In contrast, an employee of a broadcasting company looking for the time stamps within a radio programme where a particular jingle was played, would probably be looking for somewhat like an 'exact' match. In our work we deal with the latter kind of queries.

In our paper, we describe a system for full-text indexing of and searching in digital audio data. Given a short excerpt of

a waveform signal as a query, the index returns all pieces in a data base containing that waveform. Additionally, the precise position of the waveform within those pieces is returned. Our new indexing techniques also allow an approximate as well as a ranked search. This makes the index robust against several signal processing operations such as lossy compression, e.g., MPEG-1 Layer II/III coding, or addition of noise.

Our paper is organized as follows. The next section introduces the ideas behind "full-text" indexing for audio documents and reviews some previously reported approaches to audio indexing and audio-based database search. In the third section we describe our system for full-text audio indexing and give an overview of the underlying techniques. The fourth section contains results of our extensive tests. In particular we consider time-requirements for the indexing and query procedures as well as storage requirements. Prior to summing up, the fifth section discusses some applications of our technology. In the last section we suggest some of the future work in this field.

FULL-TEXT SEARCH AND RELATED APPROACHES

The Search Problem

In a classical full-text search scenario, one is interested in all occurrences of a given text string within a database of text documents. To describe our approach to audio indexing, we generalize this search task to locating a small excerpt of music within a database of (many) pieces of music.

Our search problem may be easily formulated in terms of standard signal processing notation. Given a digital signal q of finite length n and a database consisting of N signals x_i, $1 \leq i \leq N$, we are looking for the set of matches

$$\{(i,t) | q = x_i[t : t+n-1]\},$$

where $x[a : b] := (x(a), x(a+1), \ldots, x(b))$, $a \leq b$. This is, (i,t) is a *match* if the query signal x occurs in the i-th piece of music within the database starting at position t.

For most applications, such an *identity search* is by far too restrictive. If our query signal x is, for example, a decoded segment of an MP3-coded piece of music, a kind of *approximate* match, $q \approx x[a : b]$ or $|q - x[a : b]| \leq \varepsilon$, where ε is an appropriate tolerance value, would be by far more useful.

As sample-by-sample comparison is frequently an expensive as well as inappropriate strategy, one often resorts to feature-based similarity measures. If $F : \mathbb{R}^n \to \mathbb{R}^m$ denotes an m-dimensional feature extraction method, a feature-based match would be a pair (i,t) such that $F(q) = F(x_i[t : t+\ell])$ for some appropriate ℓ. We shall use the shorthand notation $F_{t,k}(x) := F(x[t : t+k-1])$. An *approximate* feature-based match to a query q is a pair (i,t) s.t. $F(q) \approx F_{t,k}(x_i)$ for some k w.r.t. an appropriate similarity measure $\approx$.

It seems to be impractical or at least very expensive to evaluate $F_{t,k}(x)$ for each document i and each position t within this document. Hence, several strategies have been proposed in the literature to circumvent this problem.

A first approach would be to evaluate $F_{t,k}(x)$ for all t in an appropriate subset $I \subset \mathbb{Z}$, e.g. $k = 1024$, $I = \{0, 512, 1024, \ldots\}$ (50% overlap processing) or $k = 2048$, $I = \{0, 2048, 4096, \ldots\}$ (non-overlapping processing). If the size of the feature vectors, m, is reasonably small, this amounts in significant savings in storage requirements for the extracted feature vectors.

Querying a database of such feature vectors with a query q of size n amounts to finding $(i,t), t \in I$, s.t. $F(q) \approx F_{t,n}(x_i)$. Clearly, if $q = x_i[s : s+n-1]$ for $s \notin I$, i.e., x is not aligned with x_i's frame boundaries, this might lead to problems in finding a match:

- A match will only be found, if $F_{t,n}(x)$ and $F_{t+\delta,n}(x)$ are sufficiently similar for each δ in the magnitude of the stepsize between two adjacent frames.
- If the former requirement is fulfilled, a match resulting from $F(q) \approx F_{t,n}(x)$ only yields an approximate position t of q within the signal x.

If the size of q is larger than the segment size n used to compute the feature vectors, q itself is segmented to produce a *sequence* of feature vectors. A query may now be seen as a comparison of sequences of feature vectors. This leads to the problem of an appropriate alignment of the query sequence with the feature sequences stored in the data base. In comparing two sequences, two feature vectors should be aligned if they are sufficiently equal. Many strategies to solve such alignment problems have been proposed in the field of speech recognition, e.g., dynamic time warping.

A different approach to reduce costs in storing feature vectors is to make the feature extraction independent of the input signal's length. This is, for each signal, only one feature vector of fixed length is computed. As an example we refer to work carried out at UC Berkeley [1]. Obviously, such an approach completely loses the time-dimension of the signal and hence is not appropriate for our kind of search problem.

Feature Extraction

We now give some examples of acoustical features which have recently been proposed for use with audio indexing. We mainly concentrate on so-called low-level features which are derived from the PCM waveform in a straightforward manner. The use of higher-level features where the extraction frequently involves a complex recognition mechanism are not considered (e.g., musical notes or instrumentation).

In [2] the authors propose to extract features from the audio signals loudness, pitch, tone, bandwidth, brightness, and the mel-filtered cepstral coefficients (MFCCs). Additionally, the use of derivatives of the corresponding trajectories (e.g. the pitch trajectory) to derive features is proposed. The features are proposed to be extracted on a frame-by-frame basis yielding a time-series of feature vectors.

In [1], a single feature vector containing information derived from tonal histograms, tonal transitions, noise characteristics, volume, tempo and rhythm is extracted for each signal. The search in the high dimensional feature space is performed by using a brute-force k-nearest neighbor search. Proposed lengths of feature vectors are 12, 24, 100, 1000, and 1248.

More recently, several parties have proposed to use psychoacoustic knowledge to make the extracted features or *fingerprints* robust against common signal distortions such as lossy compression or analog transmissions [3]. Another sound fingerprinting project is [4].

Problem Formulation: Summary

Summing up, in our approach we are interested in matches of the form (i,t), identifying the piece x_i and the position t within that piece where a given query signal is found. Furthermore, we require the indexing mechanism to be invariant

against signal modifications like analog transmissions, lossy compression, bandwidth reduction, and amplification. In the following section, we illustrate the key principles of a system satisfying those requirements.

SYSTEM FOR FULL-TEXT INDEXING

Frame- and String-Based Points of View

The traditional frame-based feature extraction as described above has the properties of fixed frame positions and fixed frame-sizes. As a consequence, the features are assigned to fixed (time-) locations. Moreover, in most approaches a feature vector is assigned to *each* frame. However intuitively, rather the signal itself should determine the time positions where features are located. This is, in our opinion, an *event-based* point of view would be more appropriate.

String-based feature representations, e.g. phoneme/diphone representations from speech processing, are more adapted to this event-based view. Unfortunately, those representations do not preserve the (time-) distance between single events. The only time-based property preserved by a string-like representation is the *succession* of events, i.e., their relative ordering. Hence, also a string-based representation is not suitable for our purposes.

Key Ideas

Having in mind the latter, we state the key principles of our full-text indexing scheme.

First, the feature-extraction should be event-based. The feature's positions are adaptively determined for each signal. Features are realized by pairs consisting of *time* and *classification* information. Hence in our index we store *patterns* of the form

$$((time_i,\ classification_i))_{i\in I}.$$

Second, in the task of music identification, absolute distances between neighboring features are an important measure characterizing a particular audio signal. Hence, time-invariance will be an important issue in our feature extraction.

Third, the preprocessing to indexing *and* querying should be performed identically. This way, the risk of introducing errors because of different feature extraction methods is minimized.

Let us also stress, that we do not use particular psychoacoustic models or measures to determine our features. Such an approach has been suggested before [3]. Yet up to now and to our knowledge, few results have been published discussing those approaches. The only psychoacoustic knowledge we incorporate in our feature extraction are some implicit assumptions which facilitate a robust selection of features.

Indexing System

Fig. 1 gives an overview of the overall indexing system. The audio signal processing module (ASP) performs a preprocessing step. The preprocessing is approximately invariant to translations of the input signal. Additionally, the ASP performs a small amount of noise reduction. Possible preprocessing could consist of windowed Fourier transforms, wavelet transforms, or filter banks. Several of those preprocessing methods were evaluated during our research. Good results were obtained by a choice of optimized filter banks.

The adaptive feature extraction unit determines relevant (time-) stamps and creates features consisting of *(time, classification)*-pairs. This way, feature patterns are created. In the design of suitable features, the *feature density* is a critical issue. In our event-based approach, no features are assigned to null-signals as those signals do not carry any interesting information. Hence, passages of silence cannot be detected unless features are explicitly assigned to such passages. On the other hand, we have to take care that features *are* assigned to every "interesting" part of an audio signal. Our current set of features is adapted to general acoustic signals. In particular, the feature density has been designed to allow a signal identification from about $0.25 - 0.5$ seconds of audio. However, the feature density may be adapted according to application-specific demands. A reduced feature density implies a higher index compression. On the other hand, this parameter setting requires longer portions of an audio signal to perform the task of signal identification. An increased feature density would allow a signal identification from, e.g., 10 milliseconds of audio, but also imply higher storage requirements to store the index. However, for most applications it will probably be not necessary to achieve an identification from such a short piece of audio. In our settings, the appropriate feature extraction parameters to guarantee a sufficient feature density have been determined and validated by extensive tests on a wide variety of acoustic material. A more detailed discussion of the test material will be given in the next section.

The next step in the indexing process consists of the reduction of the extracted features and is accomplished by a feature compression unit (FCU). This unit examines the features extracted by the AFE unit and retains only the most relevant ones. In our setting, this unit is used as an additional means to regulate the feature density as discussed in the previous paragraph. The FCU as well as the AFE works in an approximately time-invariant manner, i.e., the time-information of the retained features is approximately preserved.

The extracted features are finally organized, classified, and stored using an algebraic indexing technique. This indexing is performed by an algebraic indexing unit (AIU) which has recently been developed by our group. In brief, the AIU combines techniques from classical full-text indexing [5] and computer algebra to account for the structure contained in the extracted feature patterns. This structure is thus mapped to the index data. The use of algebraic indexing is not restricted to the field of audio. Details about the algebraic indexing techniques and applications to other fields will be reported elsewhere.

Fig. 2 gives an overview of the query processing queue. The upper part of the figure shows the raw PCM signal. The signal ist first analyzed to yield a feature space representation. In the example this is illustrated by a windowed Fourier transform (second row of the figure). The adaptive feature extraction yields a pattern of candidate features as illustrated by the stars in the third row of the figure. The feature compression unit selects the most important features among those candidates (4th row of the figure). Finally, the reduced feature set is stored in the index using the algebraic indexing technique. We would like to particularly point out the event-based feature extraction as performed by the AFE unit.

Query Processing

A query is assumed to be given by an arbitrary piece of audio. However in most cases this will be only a short excerpt containing only a few seconds of an audio recording. Naturally,

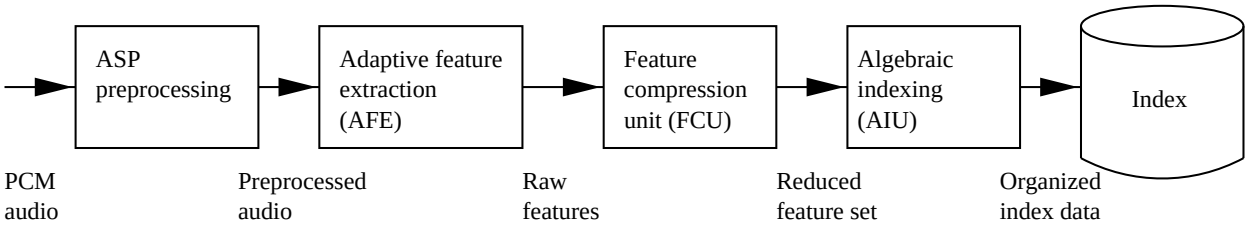

Fig. 1: Overview of the overall indexing system.

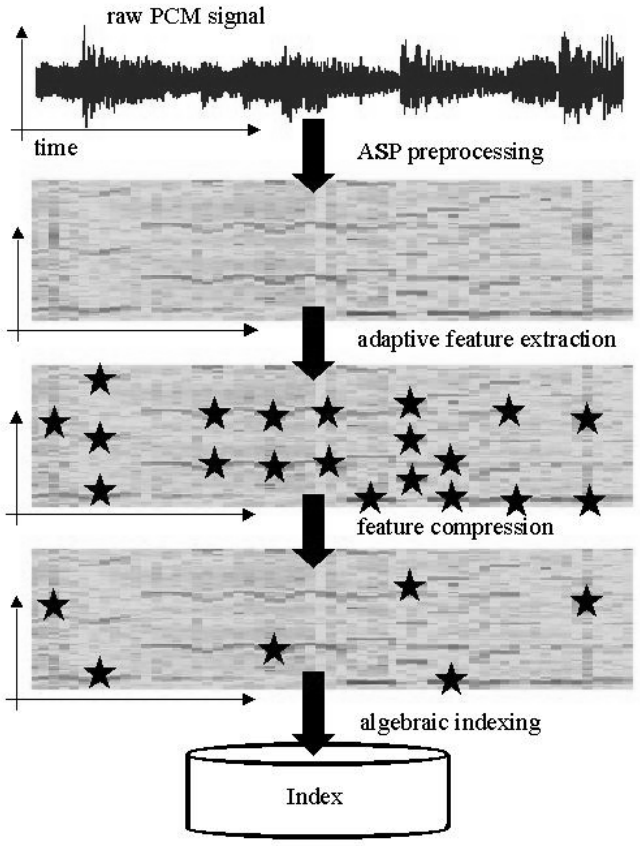

Fig. 2: Sketch of the feature extraction process. The starts in the third resp. fourth rows indicate the extracted candidate features resp. the features selected by the FCU for indexing.

the length of a query required to perform a unique identification will depend on the feature density as discussed in the last subsection.

The query is basically fed into the preprocessing queue as already described in the indexing section. ASP-processing, AFE and FC are performed in the same way as in the process of indexing. As the feature extraction is approximately time-invariant, the features extracted from a query are approximately equal to a subset of the features extracted from a piece of audio containing that query as a subpart.

The algebraic retrieval unit (ARU) searches the index for one of the following types of matches:

- *Exact matches*, i.e., the features of the query coincide with a subset of the features of an audio document within the index.
- *k%-matches*, i.e., the queries features exhibit a $k\%$-match with a part of an audio document contained in the index.
- *Approximate matches*, i.e., the features are allowed to be slightly distorted as compared to those of an audio document within the index.

The latter two types of matches provide robustness against signal distortions resulting from various kinds of signal processing as mentioned above. In our query framework it is also possible to efficiently search for combinations of those two types of matches, i.e., approximate $k\%$-matches.

PERFORMANCE

For our performance measurements we created several indexes using both a large archive of popular, rock, dance, and classical music and recordings of radio programmes of a german radio station. The main index consists of 930 rock and pop songs. The raw PCM material has a data volume of approximately 50 gigabytes. The resulting index size is 58 megabytes. That means a data reduction of approximately $1:862$. In this particular index, the left and right channels were both processed independently. Using a downmix to one mono channel or even restricting the index to only one channel, the index size can be further reduced to half of its size resulting in a total data reduction of $1:1725$. Further experiments indicate that the use of an additional index compression, e.g. by Huffman- or similar coding techniques can further reduce the index size by a factor of two. However in this case, care has to be taken not to slow down the query times because of the necessity of an additional decoding step prior to querying. Mechanisms for a suitable index compression have already been devised and will be reported elsewhere.

In summary, the size of the index possible within our current feature-settings can be estimated to roughly 1/3500-th of the size of the raw PCM data. As an illustration, imagine that approximately 3500 CDs could be indexed using *one* index CD only.

Feature Density

As discussed before, the feature density influences the minimum length of a query that is required for an identification of that query. Fig. 4 shows the number of features per processing frame for the song "Belfast Child" by the *Simple Minds*. Each processing frame corresponds to roughly 1.1 seconds of audio (49152 samples at a sampling rate of 44.1 kHz). The most frequent feature densities in this example range from $5-30$ features per second. Fig. 5 shows the feature density for a piece of orchestral music ("Star Wars Suite"). The lower part of the figure depicts the waveform signal. Assuming a feature density of $5-10$ features per second to be sufficient for a unique identification of one second of audio in a mid-size audio data base (of perhaps several 100,000 songs), the figures show our features to yield a suitable feature density.

Fig. 6 shows the indexing data for an entire audio CD ("Innuendo" by *Queen*). As in our main index, the left and right channels were indexed separately. The rightmost column shows the number of features per piece. Note that although we cannot guarantee a specific feature density per frame or per piece in our event-based setting, our features were chosen suitably to yield a very regular density. Fig. 7 shows the index size in kB per hour of indexed radio material. The radio material was directly recorded from antenna using a conventional radio receiver and PC sound recording equipment. In this case we indexed one mono channel only. The figure exhibits a very close to linear increase in the index size with time. As the size of our indexes can be proven to grow linearly with the number of indexed features, this also documents a nearly constant feature density.

Indexing performance

For our runtime experiments we used a PC with MS-Windows 98 as an operating system, 900 MHz Athlon CPU and 512 MB of main memory. Note however that our indexing and query engines have been designed for much larger data bases than presented in our tests. In our implementation, the index size is *not* limited by the available main memory. Furthermore it is not necessary to load the whole index into the computers main memory prior to query processing. For our main test index of size 58 MB, a standard PC with only 16 MB of main memory would be sufficient for the query processing.

The process of indexing PCM audio turns out to be realizeable several times faster than realtime. For indexing one CD of 53:49 minutes (543 MB) of audio, the index construction takes 10:42 minutes when both channels are indexed separately. This is roughly a factor of 5 faster than realtime. When restricting the indexing to one channel only, the indexing speed is doubled. The indexing of a second audio CD of length 40:54 minutes ("Blues Brothers Soundtrack") (413 MB of data) takes 8:23 minutes yielding similar runtime/realtime ratios.

Query Performance

Fig. 8 shows the query performance data of 16 short audio pieces we used for querying or main index. All pieces are of short lengths ranging from $0.5-3$ seconds. The third column from the left gives the total query times. The rightmost column gives the number of features extracted from each of the queries. In the setting of an exact matching, all queries are uniquely identified, i.e. each of them yields an exact match of the form (i,t) consisting of the identified piece i and position t within that piece where the query occurs. The response times to queries are very short and range from $0.5-1.5$ seconds.

Robustness of Features

To allow a signal identification from distorted versions of an original audio signal, the features should exhibit a certain amount of robustness. Ideally, the features of the original

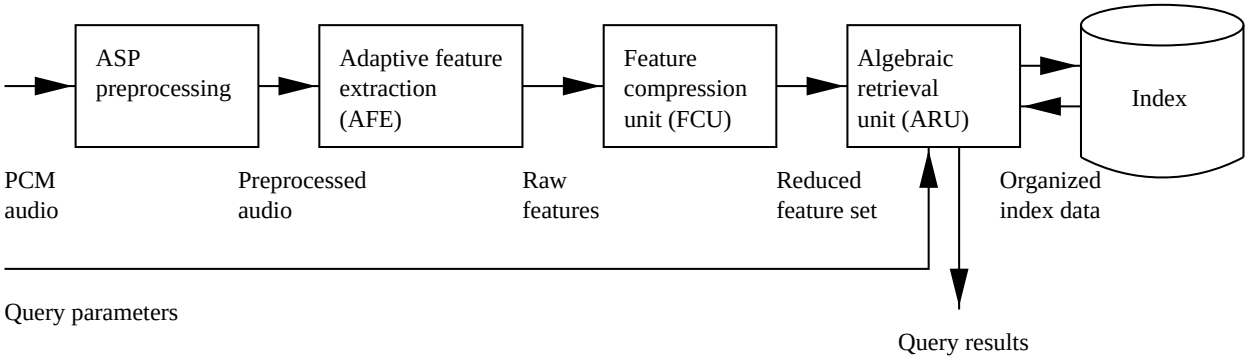

Fig. 3: Overview of the query processing.

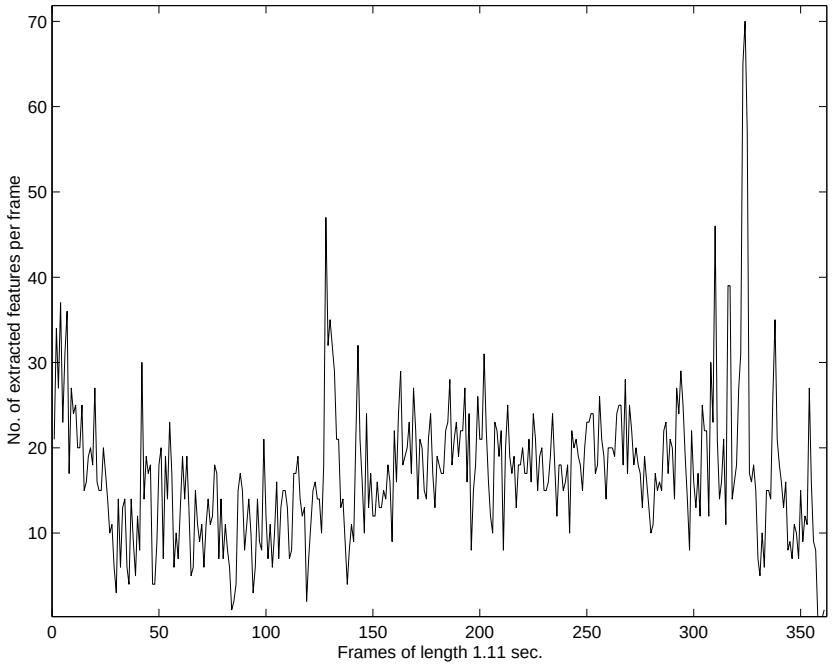

Fig. 4: Feature density of the left channel of the song "Belfast Child" (Simple Minds).

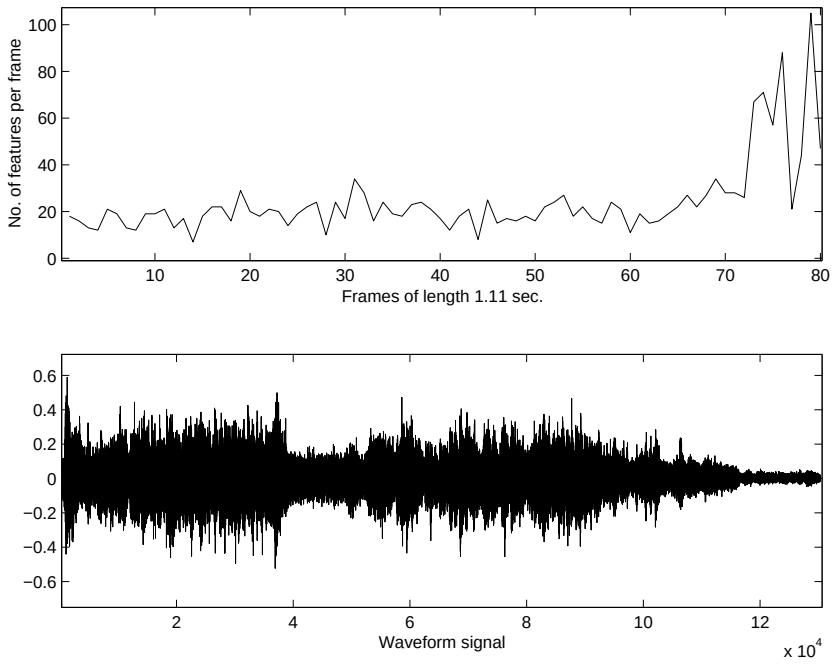

Fig. 5: Feature density of the left channel of the first 1:30 min. of the "Star Wars Suite" played by the Collegium Musicum, University of Berlin, Germany.

signal and a distorted but yet perceptually identifiable signal should be the same. As there are many forms of relevant signal distortions, e.g., lossy compression or analog-, noisy-, or radio transmissions, which themselves might result in very different degrees and types of distortion, this goal can only approximately be achieved. Moreover, the notion of "perceptually identifiable" is a very subjective one and cannot be expressed by simple formulae.

Observing our feature patterns of the form

$$((time_i,\ classification_i))_{i\in I}.$$

we consider two types of possible changes in the query pattern resulting from signal distortions as compared to a pattern stored in the index:

- The time-parameter of single features could be modified or slightly distorted. For this kind of distortion we allow a user-configurable tolerance value. This value may be freely chosen depending on the assumed degree of signal distortion. It determines the maximum time-deviation between two features with identical classifications that is allowed to consider both to be identical.
- The classification-parameter of a feature could change as a result of a signal distortion. As we generally do not assume the existence of a distance measure between two different classifications, such differences in the classification parameter are considered to yield a *feature mismatch*. To make our query scheme robust against distortions causing mismatches we allow a certain percentage of feature mismatches for each query pattern. This percentage may be predefined by a user, e.g., 40% of feature mismatches could be specified to be tolerable for a feature pattern to still yield a match.

Fig. 9 illustrates our concept of robustness comparing the features of an MP3-coded version of a signal to those of a corresponding original signal. The comparison is performed on a segmental basis: The MP3-coded signal is decoded and the features are extracted using the ASP, AFE and FC units. Subsequent segments of 30 features each are grouped together and used as a query input to a data base containing the indexed original signal. The figure for each segment shows the percentage of features coinciding in the original and the MP3-version of the signal. In this setting, a tolerance value of 50 samples was allowed as an additional time-deviation for each feature. Note that with the exception of two segments the percentage of coinciding features is very high. On the other hand, our tests show that *false matches*, i.e., accidental coincidents of a high percentage of features per segment, occur very rarely. In our tests, all of those false matches could be resolved by examining two or three segments with high percentage matches. Intersecting the matches for those segments, thus only retaining the matches occurring in all of those segments, generally yields correct identifications.

The MP3-coding used in our tests was performed using a compression ratio of $1:12$. Put it another way, only roughly 8% of the original signal is conserved. In our setting of radio-indexing, a considerable extent of the recorded audio material contains transmission distortions. However it turns out that

Piece no.	Channel	Length (samples)	Length (min.)	No. of features
1	left	9,665,291	3:39	3,983
1	right	9,665,291	3:39	3,925
2	left	11,563,787	4:22	4,119
2	right	11,563,787	4:22	4,070
3	left	11,264,267	4:15	3,798
3	right	11,264,267	4:15	3,706
4	left	11,539,595	4:22	4,633
4	right	11,539,595	4:22	4,710
5	left	12,163,979	4:36	5,445
5	right	12,163,979	4:36	5,349
6	left	9,567,371	3:37	5,456
6	right	9,567,371	3:37	5,285
7	left	12,479,627	4:43	3,677
7	right	12,479,627	4:43	3,692
8	left	13,082,123	4:57	5,336
8	right	13,082,123	4:57	5,194
9	left	17,258,123	6:31	6,819
9	right	17,258,123	6:31	6,907
10	left	12,066,059	4:34	4,918
10	right	12,066,059	4:34	4,907
11	left	12,271,115	4:38	4,349
11	right	12,271,115	4:38	4,434
12	left	9,499,403	3:35	3,281
12	right	9,499,403	3:35	3,217

Fig. 6: Indexing data for an entire CD ("Innuendo" by *Queen*).

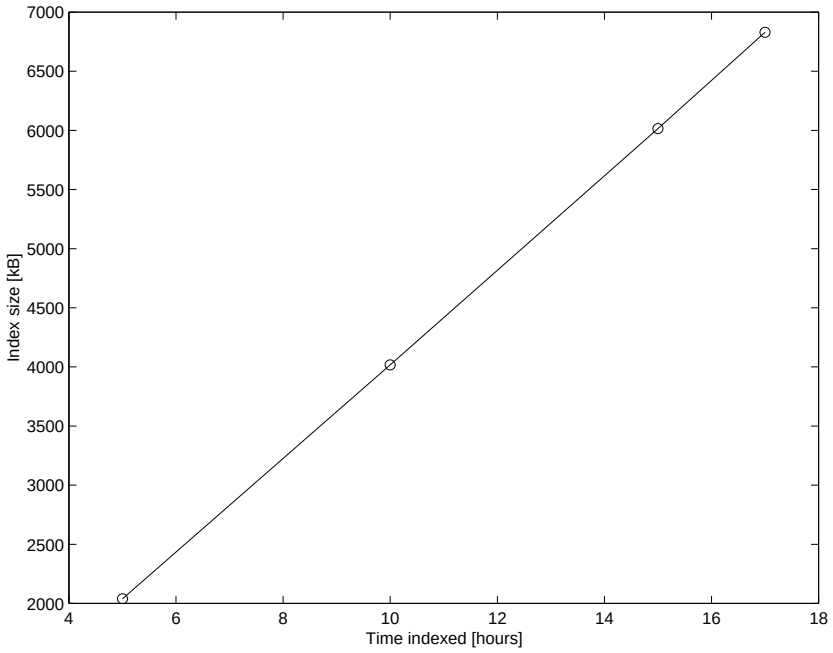

Fig. 7: Index size obtained by indexing several hours of german radio (mostly pop/rock music and speech).

Piece no.	Length in seconds	Query time in seconds	No. of features
1	0.565	0.38	5
2	0.491	0.71	10
3	0.636	0.66	8
4	1.829	0.72	12
5	1.142	1.04	18
6	0.801	0.5	9
7	1.141	1.1	16
8	1.161	1.59	25
9	1.157	1.09	14
10	0.925	0.6	11
11	0.641	0.66	8
12	0.957	1.27	17
13	0.833	0.82	13
14	1.115	0.88	14
15	1.12	0.71	15
16	2.98	1.43	28

Fig. 8: Query times for exact identifications on the test data base containing 50 GB of indexed PCM audio.

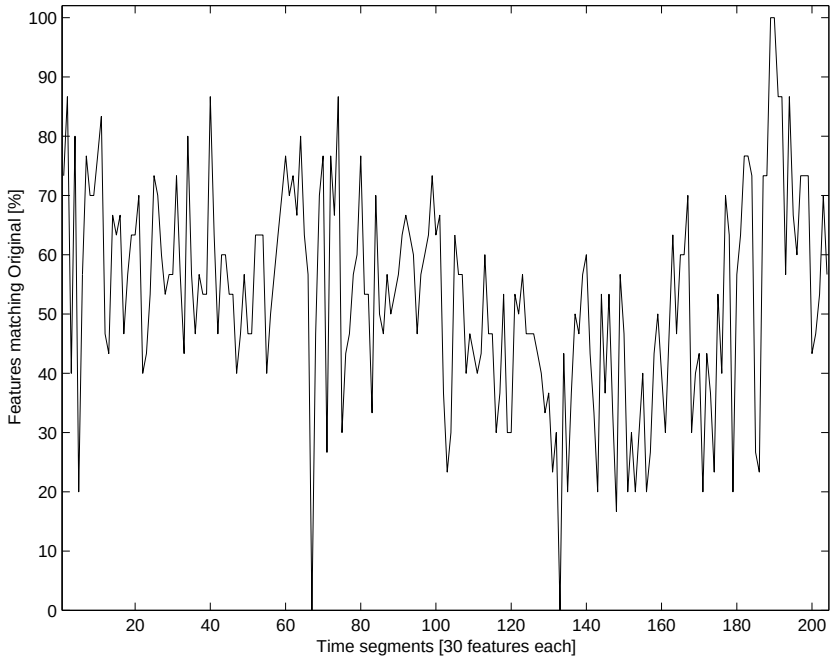

Fig. 9: Robustness of our features against MP3-compression. Depicted is the percentage of features for each time segment which coincide in the MP3-coded version and the original signal.

although those distortions are partially severe, a reliable signal identification is still possible from our chosen features.

Let us stress at this point that the set of features used in our current settings has not explicitly been designed to provide a robustness against higher level distortions. The design of such features is beyond the scope of this work and will be addressed in the future.

APPLICATIONS

There are several areas of application for our novel indexing technique. To describe these areas, note that we have made a clear distinction between *feature extraction* resp. preprocessing and *indexing*. The only requirement for the feature extraction is that it should produce patterns of the form $((time_i,\ classification_i))_{i\in I}$. This form exhibits the essential characteristics of our event-based approach.

As our approach is very modular, it may be easily combined with any suitable feature extraction method. The chosen features determine how precise a given audio document is represented in the index. In our setting of full-text indexing we look for exact- or indentity-matches. However, several other projects in the field of audio retrieval aim at *similarity*-based matches. Various types of similarity have been discussed recently. The features used in those approaches are mostly higher level features such as instrument classes or note onsets. A straightforward application of our technology is to combine the feature extraction methods used by those approaches with our event-based view and the algebraic indexing technique.

Applications of audio identification as discussed in this paper have recently been brought up by the industry as well as the research community:

- Automated monitoring of radio and TV programmes, particularly automatic detection of music, ads, audio clips, etc.
- Monitoring of audio traffic on the internet. The faster than realtime capabilities of our retrieval technique allow the detection of copyrighted material during its transmission. Furthermore, only fractions of the audio material (some seconds or few IP packets) are needed for an identification.
- Detection of audio material illegally distributed on the internet.
- Identification of audio fragments played back by a user querying for the CD/DVD containing the corresponding audio track.
- Directly querying PCM/MP3-audio instead of only the title, composer, etc. (classical vs content-based queries).

The applications of the very powerful algebraic indexing technique (AIT) used by our full-text audio indexing are in many fields. Given a suitable feature extraction depending on the application specific demands, this technique allows for very fast retrieval of complex query patterns in very large data sets. In our group we are currently developing a system for the fast search of objects in complex 3D scenes.

SUMMARY AND FUTURE WORK

In our paper we proposed a novel event-based framework for full-text audio indexing. The main application described in this paper is the *identification* of pieces of audio resp. the index-based search of a short piece of audio in a very large audio data base. Our indexing method combines an algebraic indexing technique with a suitable feature extraction. For PCM audio the index is compressed to factors of $1:1000$ and above w.r.t. the original PCM data. The indexing is robust to mid-level signal distortions such as MP3 compression or radio transmissions.

Our future work in the field of audio will investigate suitable high-level features to make the indexing robust to more severe kinds of signal distortions. The algebraic indexing techniques will be applied to other multimedia data as indicated above.

REFERENCES

[1] Matt Welsh, Nikita Borisov, Jason Hill, Robert von Behren, and Alec Woo. Querying Large Collections of Music for Similarity. Technical report, Computer Science Division, University of California, Berkeley, 1999.

[2] Erling Wold, Thom Blum, Douglas Kreislar, and James Wheaton. Classification, search, and retrieval of audio, 1999. CRC Handbook of Multimedia Computing.

[3] Tuneprint. Robust Psychoacoustic Fingerprinting, 2000. `http://www.tuneprint.com/`.

[4] eTantrum. eTantrum Music ID, 2000. `http://www.etantrum.com/`.

[5] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes*. Van Nostrand Reinhold, 2nd edition, 1999.