Efficient Index-Based Audio Matching

Article *in* IEEE Transactions on Audio Speech and Language Processing · March 2008

DOI: 10.1109/TASL.2007.911552 · Source: IEEE Xplore

Meinard Müller
Friedrich-Alexander-Universität of Erlangen-Nürnberg
392 PUBLICATIONS **12,534** CITATIONS

Efficient Index-Based Audio Matching

Frank Kurth, *Member, IEEE*, and Meinard Müller

***Abstract*—Given a large audio database of music recordings, the goal of classical *audio identification* is to identify a particular audio recording by means of a short audio fragment. Even though recent identification algorithms show a significant degree of robustness towards noise, MP3 compression artifacts, and uniform temporal distortions, the notion of similarity is rather close to the identity. In this paper, we address a higher level retrieval problem, which we refer to as *audio matching*: given a short query audio clip, the goal is to automatically retrieve all excerpts from all recordings within the database that *musically* correspond to the query. In our matching scenario, opposed to classical audio identification, we allow semantically motivated variations as they typically occur in different interpretations of a piece of music. To this end, this paper presents an efficient and robust audio matching procedure that works even in the presence of significant variations, such as *nonlinear* temporal, dynamical, and spectral deviations, where existing algorithms for audio identification would fail. Furthermore, the combination of various deformation- and fault-tolerance mechanisms allows us to employ standard indexing techniques to obtain an *efficient, index-based* matching procedure, thus providing an important step towards semantically searching large-scale real-world music collections.**

***Index Terms*—Audio indexing, audio matching, chroma features, music retrieval, musical interpretation, work identification.**

I. Introduction

EVEN though there is a rapidly growing corpus of audio material, there still is a lack of efficient systems for content-based audio retrieval, which allow the user to explore and browse through large music collections without relying on manually generated annotations. In this context, the query-by-example paradigm has attracted a large amount of attention: given an excerpt of an audio recording (a *query* clip), the task is to automatically retrieve all excerpts from a given music database containing parts or aspects similar to the query. Here, the notion of similarity used to compare different audio clips is of crucial importance and largely depends on the respective application as well as the user requirements.

In the scenario of *audio identification*, the retrieval task is to identify a particular audio recording, which is assumed to be part of an audio database, and to deliver the title, composer, or artists of the underlying piece. In an extended form, the identification problem also includes the task of temporally locating the query clip within the original audio recording. Even though recent identification algorithms show a significant degree of robustness towards noise, MP3 compression artifacts, and uniform temporal distortions [1], [2], the notion of similarity used in the scenario of audio identification is rather close to the identity. Existing algorithms for audio identification cannot deal with strong nonlinear temporal distortions or with other musically motivated variations that concern, for example, the articulation or instrumentation. In other words, audio identification is a low-level retrieval task in the sense that only a *specific* audio recording (the one containing the query clip) is to be identified.

While the problem of audio identification can be regarded as largely solved even for large-scale music collections, semantically more advanced retrieval tasks are still mostly unsolved. In this paper, we address a higher level retrieval problem, which we refer to as *audio matching*: given a short query audio clip, the goal is to automatically retrieve all excerpts from all recordings within a given audio database that *musically* correspond to the query. In our matching scenario, opposed to classical audio identification, we allow semantically motivated variations as they typically appear in different interpretations and arrangements of a piece of music. For example, two interpretations may exhibit significant nonlinear global and local differences in tempo, articulation, and phrasing as well as variations in executing ritardandi, accelerandi, fermata, or ornamentations. Furthermore, one has to deal with considerable dynamical and spectral deviations, which are due to differences in instrumentation, loudness, tone color, accentuation, and so on.

To further illustrate our audio matching scenario, we give two representative examples. As a first example, the audio query clip may consist of the first 20 measures of Beethoven's Fifth Symphony in an interpretation by Leonard Bernstein. Then the goal of audio matching is to find all other corresponding audio clips in the database; this includes the repetitions in the exposition and in the recapitulation of the same interpretation as well as the corresponding excerpts in all recordings of the same piece interpreted by other conductors such as Herbert von Karajan or even in some piano arrangement or a synthesized version of a corresponding MIDI file. For Beethoven's Fifth Symphony, there are numerous interpretations (much more than 100 audio recordings are commercially available) with significant relative tempo differences of up to 25%. As a second example, the audio query may consist of the theme of Ravel's Bolero. Then, the audio matching procedure should detect the recurrent theme in the Bolero, which is played in different instrumentations including flute, clarinet, bassoon, saxophone, trumpet, strings, and culminating in the full orchestra, while the volume gradually grows

Manuscript received December 11, 2006; revised September 18, 2007. This research has been carried out while both authors were with the Multimedia Signal Processing Group, Department of Computer Science III, University of Bonn, Bonn, Germany. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Malcolm Slaney.

F. Kurth is with the Research Institute for Communication, Information Processing and Ergonomics (FKIE), Research Establishment for Applied Scenice (FGAN), 53343 Wachtberg, Germany (e-mail: kurth@fgan.de).

M. Müller is with the Max-Planck Institut für Informatik, Department D4—Computer Graphics, 66123 Saarbrücken, Germany (e-mail: mmueller@mpi-inf.mpg.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASL.2007.911552

from quiet pianissimo to a vehement fortissimo. Such types of variations typically occur for classical music and, to a lesser degree, for popular music.

Existing algorithms for audio identification do not work in the presence of the above described variations. In this paper, we propose an efficient and robust method for audio matching that can, to a large degree, cope with such variations. As we will show, our approach works well for Western music based on the equal-tempered scale including genre such as classical, popular, rock, and folk music. In Section I-A, we describe the contributions of this paper in more detail and give an overview of the paper's organization. Then, in Section I-B, we discuss related work.

A. Contributions

One general principle to cope with data variations is to extract relatively coarse features from the raw data that closely correlate to semantic aspects of the underlying data while showing a high degree of invariance to irrelevant deformations and variations. In the music context, chroma-based audio features have turned out to be a powerful tool [3]. Chroma features closely correlate to the short-time harmonic content of the audio signal and show a high degree of invariance to changes in instrumentation and tone color. In this paper, we describe how to further process the chroma features in order to introduce a higher degree of invariance. First, we temporally blur these features in order to achieve robustness to local tempo variations. Second, we normalize the resulting features (obtaining what we call *chroma energy normalized statistics (CENS) features*) to achieve invariance to deviations in dynamics, which is needed to cover a wide dynamical range as illustrated by the above Ravel example. Audio matching can then be performed on the feature level by basically comparing the query feature sequence to any feature subsequence (of the same length) of the database. This procedure, which is referred to as *diagonal matching*, has been proposed in [4] and will be summarized in Section II. Furthermore, to cope with more global tempo variations and global transpositions (pitch shifts), we employ the concept of multiple queries. Here, we simulate the tempo and pitch variations on the feature level resulting in several query feature sequences, which are then processed simultaneously. Finally, we use a local distances measure that can be expressed in terms of the inner vector product. This allows us to perform the diagonal matching efficiently by using fast convolution techniques.

Diagonal matching linearly depends on the size of the database. As a further contribution of this paper, we present a novel index-based matching strategy, which significantly speeds up the retrieval process while obtaining competitive retrieval results. To this end, we describe two methods for deriving semantically meaningful feature codebooks, which are used for quantizing the CENS audio features (Section III). According to the assigned codebook vector, the features can then be stored in some inverted file index—a well-known index structure that is frequently used in standard text retrieval.

In Section IV, we then describe our efficient index-based audio matching procedure. In particular, we introduce and combine various fault tolerance mechanisms such as fuzzy search, mismatch search, and the strategy of using multiple queries. Note that it is the *combination* of employing deformation-tolerant features and fault tolerance mechanisms (to cope with more local variations) and the strategy of using multiple queries (to cope with more global variations) that enables us to use rather inflexible but highly efficient indexing techniques even in the presence of significant nonlinear temporal and spectral variations. This is a major progress with respect to previously described index-based algorithms for the task of audio identification, which cannot handle such variations. Furthermore, we discuss a two-stage ranking strategy, which allows us to suppress false positive matches in the first stage and to produce a ranked list of high-quality audio matches in the second stage.

To demonstrate the practicability and efficiency of our novel matching algorithm, we report on our experimental results, which have been conducted on two data sets (Section V). One data set consists of more than 1000 pieces of classical music (corresponding to 112 h of audio material). This data set is versatile (e.g., it contains organ pieces by Bach, orchestral works by Beethoven, percussive pieces by Bartok, vocal music by Orff) and comprises various subgenre, which show a large variety in instrumentation, style, tempo, or dynamics. To demonstrate applicability to other genres, we have additionally evaluated our matching procedure on a data set consisting of 500 pieces of popular, hardrock, rock, and folk music. As it turns out, our approach works well for any Western music that is based on the equal-tempered scale. Furthermore, experimental results show that our proposed index-based matching approach reduces the retrieval time (in comparison to diagonal matching) by a factor of more than 15—in our current MATLAB implementation running on a standard PC it takes less then a second to retrieve a ranked list of matches from the 112-h database.

We conclude this paper with Section VI. In particular, we show how audio matching can be used not only for retrieval applications but also for convenient inter- and intra-document browsing in large music collections, see Figs. 7 and 8.

B. Related Work

We now discuss related work and give references to the literature. As we mentioned earlier, the problem of *audio identification*, which is also sometimes referred to as *audio fingerprinting*, has been studied extensively and can be regarded as largely solved, see, e.g., [1], [2], [5]–[9]. The indexing approach used in this paper adopts existing techniques for audio identification based on inverted files, initially proposed in [10] and subsequently improved in [6], [11]. In contrast to hashing strategies such as [5] that rely on multidimensional (binary) features to match hash signatures stored in an index, inverted-file-based approaches only require a particular percentage of features in some local temporal neighborhood of a query signal to match the features stored in a suitable index.

The approach to audio identification proposed by Wang [8] is more robust than hashing in that, similar to our inverted file approach, only a small fraction of matching features is sufficient to perform successful identification. While the latter is robust against severe signal distortions, the features used in [8] are on a rather physical level. Hence, those features are not invariant against distortions resulting from changes in musical parameters such as tempo, timbre, or pitch. The matching algorithm, in essence, consists of a cross correlation of the query features with those stored in the index structure. Although this kind of matching may be performed very efficiently using hash-tables and parallel processing on multiple computers, as a consequence of the rigid correlation-like approach, this method is not capable of tolerating local temporal variations as it would be necessary when applying it to audio matching. By combining suitable chroma-based features with a multiple querying strategy and several fault tolerance mechanisms, our subsequently proposed approach is capable of overcoming this problem.

Even though existing algorithms for audio identification show a high degree of robustness towards noise, MP3 compression artifacts, and uniform temporal distortions, they cannot handle nonlinear temporal distortions or variations that concern, for example, the articulation or instrumentation. Thus, those algorithms are incapable of performing audio matching as proposed in this paper.

Our audio matching scenario differs from the problem referred to as *audio alignment* or *audio synchronization*, see, e.g., [12]–[16]. (Note that in [14], the authors use the term *audio matching* in the sense of *audio alignment*.) In *audio alignment*, the goal is to globally align two explicitly given, entire audio recordings of the same underlying piece. Opposed to this, the goal of *audio matching*, as we understand it, is to identify and retrieve musically related audio fragments (typically 10–40 s of length), which are hidden in some large audio collection. Note that for the latter problem, there may be even several matches within one and the same recording.

Another related scenario is the recently studied problem of *cover song identification*, where the goal is to identify different versions of the same piece of music within a database, see, e.g., [17]–[20]. A cover song may differ from the original song with respect to instrumentation and harmony, it may represent a different genre, or it may be a remix with a different musical structure. Different techniques have been suggested to deal with temporal variations including correlation and DTW-based methods [21], beat tracking methods [20], and audio shingles (small chunks of audio) [17], [18]. Note that there is a fundamental difference between cover song identification and audio matching: in cover song identification the objective is to assign one similarity value that expresses the relatedness of *entire* audio recordings of pieces or songs, which can then be used for music classification and recommendation tasks [22], [23]. In audio matching, however, the objective is to compute similarity values for *all subsegments* of audio recordings that are related to a given query audio fragment. Such information can then be used, e.g., for efficient navigation within a single CD recording (intra-document navigation) or between different performances (inter-document navigation), see Figs. 7 and 8.

Finally, we give some references regarding techniques that will be used in our audio matching procedure. Further references will be given in the respective sections. Chroma-based audio features have recently become very popular in the MIR community with applications in various domains such as audio pattern discovery [24], audio thumbnailing and chorus detection [3], [25], [26], or audio alignment [12], [14], [15]. The reduction of high-dimensional features to small codebooks is used for a wide range of applications in the domain of speech and audio processing. For example, vector quantization of audio features based on K-means has been used for audio identification [1] and audio analysis [27]. Inverted file indexing is a standard technique in text retrieval [28], which has also been applied in the music context, e.g., for music identification [11]. In the present paper, we will use a similar technique as a basis for efficient audio matching. In our retrieval technique, the temporal order in which the extracted features occur plays a fundamental role in identifying related audio clips. The importance of using contextual information by considering feature sequences (rather then disregarding their temporal context) in determining musical similarity has also been emphasized in [27], [29].

II. Audio Matching

In this section, we describe the audio matching procedure that constitutes the basis for our fast index-based matching algorithm. The technique has been first described in [4]. The main ingredients of this procedure are the usage of coarse chroma-based audio features that absorb variations in timbre, dynamics, and local tempo (Section II-A) as well as a robust matching procedure that handles global variations in tempo and musical key (modulation) (Section II-B).

A. CENS Features

In the audio matching scenario, the notion of similarity used to compare the audio material is of crucial importance. Particularly, we are interested in identifying and retrieving audio clips irrespective of temporal and spectral differences that are typically present in different interpretations and arrangements of the same underlying musical work. Therefore, the idea is to use very coarse audio features that strongly correlate to the harmonic progression of the audio signal while showing a high degree of robustness to variations in parameters such as timbre, dynamics, articulation, and local tempo deviations.

In this context, chroma-based audio features have turned out to be a powerful mid-level representation in the music retrieval context, where the chroma correspond to the twelve traditional pitch classes $C, C^\sharp, D, \ldots, B$ of the equal-tempered scale, see [3]. In the first step, we convert the audio signal into a sequence of 12-dimensional chroma vectors. Let $v = (v(1), v(2) \ldots, v(12)) \in \mathbb{R}^{12}$ denote such a vector, then each entry expresses the short-time energy content of the signal in the respective chroma, where $v(1)$ corresponds to chroma C, $v(2)$ to chroma $C^\sharp$, and so on. Such a chroma decomposition can be obtained in various ways, for example by suitably pooling the spectral coefficients obtained from a short-time Fourier transform (STFT) [3] or by using suitable multirate filter bank techniques [30]. For details, we refer to the cited literature. Due to the octave equivalence, chroma features show a high degree of robustness to variations in timbre and instrumentation. In the following, we will use a temporal feature resolution of 10 Hz, where each chroma vector corresponds to a window of 200 ms with a window overlap of half the size. As an example, Fig. 1(a) shows the chroma representations of two recordings of the first 21 measures of Beethoven's Fifth—an orchestral version conducted by Bernstein and a piano arrangement played by Scherbakov.

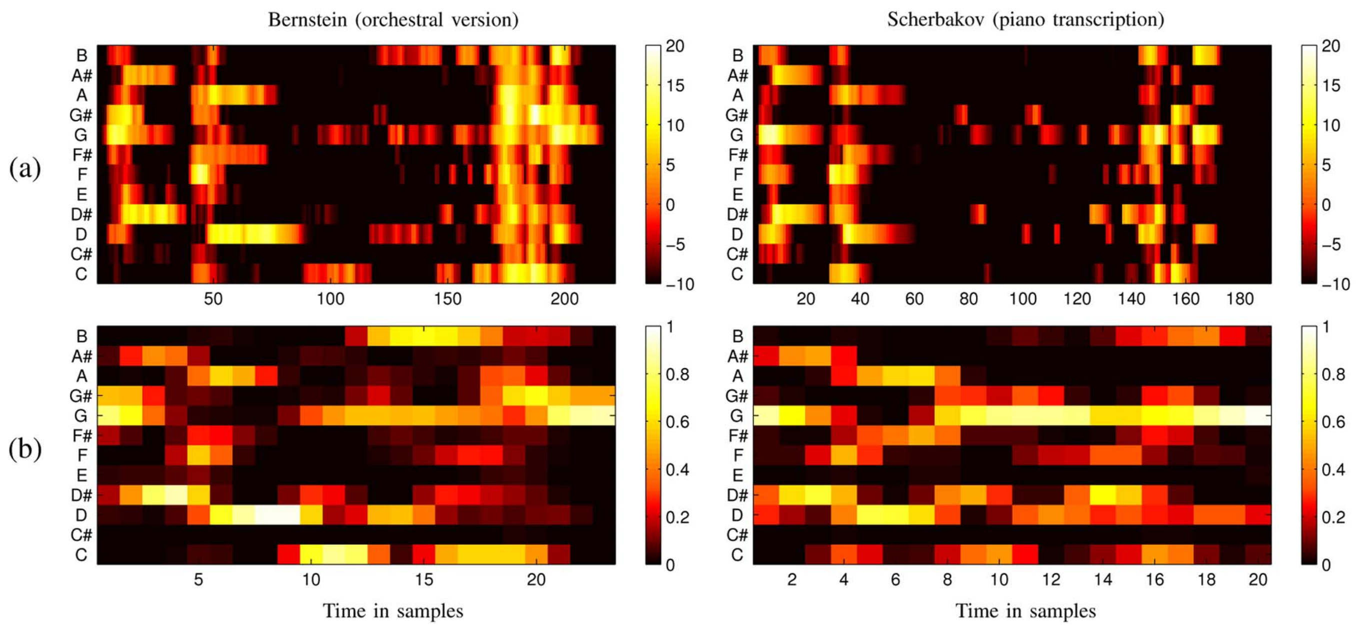


Fig. 1. (a) Chroma representations (10 features per second) of two recordings of the first 21 measures of Beethoven's Fifth Symphony. Left: Orchestral version conducted by Bernstein (22 s). Right: Piano reduced version played by Scherbakov (19 s). (b) Corresponding sequences of CENS$_{10}^{41}$-features (1 feature per second). The CENS representation more clearly reveals the similarity of both versions.

To account for robustness to the other parameters listed above, we now introduce several modifications of the chroma representation. To absorb variations in dynamics, each chroma vector v is replaced by $v/\sum_{i=1}^{12} v(i)$, which expresses the relative distribution of the signal's energy within the 12 chroma bands. To avoid random energy distributions occurring during passages of very low energy (like passages of silence before the actual start of the recording or during long pauses), we use a simple energy-thresholding method to replace a chroma vector v by the uniform distribution in case v is close to the zero vector. Next, we consider two types of short-time statistics over these energy distributions. First, we introduce a quantization function $\tau : [0,1] \to \{0,1,2,3,4\}$ defined by $\tau(a) := 0$ for $a \in [0,0.05)$, $\tau(a) := 1$ for $a \in [0.05,0.1)$, $\tau(a) := 2$ for $a \in [0.1,0.2)$, $\tau(a) := 3$ for $a \in [0.2,0.4)$, and $\tau(a) := 4$ for $a \in [0.4,1]$. Then, each chroma energy distribution vector $v = (v(1),\ldots,v(12)) \in [0,1]^{12}$ is quantized by applying τ to each component of v, yielding $\tau(v) := (\tau(v(1)),\ldots,\tau(v(12)))$. Note that the thresholds are chosen in a logarithmic fashion to account for the logarithmic sensation of sound intensity, see [31]. In the second step, the sequence of quantized chroma distribution vectors is convolved component-wise with a Hann window of length $\mathbf{w} \in \mathbb{N}$ and then downsampled by a factor of $\mathbf{d} \in \mathbb{N}$. This again results in a sequence of 12-dimensional vectors, which are finally normalized with respect to the Euclidean norm. The resulting features are referred to as CENS$_{\mathbf{d}}^{\mathbf{w}}$ (chroma energy normalized statistics), which represent a kind of weighted statistics of the energy distribution over a window of $\mathbf{w}$ consecutive vectors. Note that these features are elements of the set

$$\mathcal{F} := \left\{ v = (v(1),\ldots,v(12)) \in [0,1]^{12} \mid \|v\|_2 = 1 \right\} \qquad (1)$$

where $\|v\|_2 := \left(\sum_{i=1}^{12} v(i)^2\right)^{(1/2)}$. In other words, $\mathcal{F}$ consists of all points on the unit sphere S^{11} that have nonnegative entries. For later use, we define a projection operator $\pi^{\mathcal{F}} : \mathbb{R}^{12} \to S^{11}$ by

$$\pi^{\mathcal{F}}(v) := \begin{cases} \frac{v}{\|v\|_2}, & \text{if } \|v\|_2 \neq 0 \\ \frac{(1,1,\ldots,1)}{\sqrt{12}}, & \text{if } \|v\|_2 = 0. \end{cases} \qquad (2)$$

The main idea of CENS features is that taking statistics over relatively large windows not only smooths out local time deviations as they may occur for articulatory reasons but also compensates for different realizations of note groups such as trills or arpeggios. As an example, consider the sequences of CENS$_{10}^{41}$-features for our Beethoven example, see Fig. 1(b). Even though the two interpretations exhibit significant variations in articulation and instrumentation, the resulting CENS sequences are similar. Finally, note that one obtains a flexible and computationally inexpensive procedure to adjust the feature resolution by simply modifying the parameters $\mathbf{w}$ and $\mathbf{d}$ in the CENS computation (without repeating the cost-intensive spectral audio decomposition). This will be exploited in Section III to efficiently simulate tempo changes.

B. *Diagonal Matching*

In our setting, an audio database consists of a collection of CD recordings, typically containing various interpretations for one and the same piece of music. To simplify things, we may assume that this collection is represented by one large document D by concatenating the individual recordings and keeping track of the boundaries in a supplemental data structure. In a preprocessing step, D is transformed into a sequence of CENS$_{10}^{41}$-features denoted by CENS$_{10}^{41}[D] = (w_0, w_1, \ldots, w_{M-1})$.

For the audio matching task addressed in this paper, a typical query Q is an audio clip of a duration of 10–40 seconds. This range covers typical durations of musically relevant passages such as themes or parts of chorus sections within audio recordings. In the basic matching procedure, the query Q is also transformed into a CENS feature sequence $\mathrm{CENS}^{41}_{10}[Q] = (v_0, v_1, \ldots, v_{N-1})$. This query sequence is compared to any subsequence $(w_i, w_{i+1}, \ldots, w_{i+N-1})$ consisting of N consecutive vectors of the database sequence and starting at position $i \in [0 : M-N]$. Then, a *match* of the query Q with respect to the database is defined to be a tuple (k, N) consisting of the *match position* $k \in [0 : M-1]$ and the *match length* $N \in [1 : M-1]$. We then also say that the query Q matches the audio clip corresponding to the feature subsequence $(w_k, \ldots, w_{k+N-1})$.

For comparing the query and database sequence, we use the distance measure

$$\Delta(i) := 1 - \frac{1}{N} \sum_{n=0}^{N-1} \langle w_{i+n}, v_n \rangle. \tag{3}$$

Note that since all CENS vectors are of unit length, the inner products $\langle w_{i+n}, v_n \rangle$ coincide with the cosine of the angle between w_{i+n} and v_n. Altogether, we obtain a distance function $\Delta : [0 : M-1] \to [0,1]$ by setting $\Delta(i) := \infty$ for $i \in [M-N+1 : M-1]$. As the measure (3) is computed basically by summing up diagonals of a matrix $(\langle w_m, v_n \rangle)_{mn}$, the proposed matching technique will be referred to as *diagonal matching*. To determine the best match between Q and D, we simply look for the index $i_0 \in [0 : M-1]$ minimizing Δ. Then the best match is given by the tuple (i_0, N), which encodes the audio clip corresponding to the feature sequence $(w_{i_0}, w_{i_0+1}, \ldots, w_{i_0+N-1})$. To look for the second best match, we exclude a neighborhood around the index i_0 from further consideration to avoid large overlaps with the best match. To find subsequent matches, the latter procedure is repeated until a certain number of matches is obtained or a specified distance threshold is exceeded, see Fig. 3.

Due to the temporal blurring of the CENS features, this strict matching procedure works well even in the presence of local tempo variations. However, in the presence of global tempo differences, this basic matching procedure does not yet work well, see Fig. 2. For example, Bernstein's interpretation of the first movement of Beethoven's Fifth has a much slower tempo (roughly 80%) than Karajan's interpretation. While there are 23 CENS feature vectors for the first 21 measures computed from Bernstein's interpretation, there are only 19 in Karajan's case. To account for such global tempo variations, we create several versions of the query audio clip corresponding to different tempi. These tempo changes are simulated on the CENS feature level by suitably modifying the parameters $\mathbf{w}$ and $\mathbf{d}$. More precisely, instead of only considering $\mathrm{CENS}^{41}_{10}[Q]$, we calculate feature sequences $\mathrm{CENS}^{\mathbf{w}_j}_{\mathbf{d}_j}[Q]$ of length N_j for eight different pairs $(\mathbf{w}_j, \mathbf{d}_j)$ with $\mathbf{d}_j = j + 6$ and $\mathbf{w}_j = 4\mathbf{d}_j + 1$, $j \in [1 : 8]$. Each pair effectively simulates a tempo change. For example, using $\mathbf{d}_7 = 13$ (instead of $\mathbf{d}_4 = 10$) and $\mathbf{w}_7 = 53$ (instead of $\mathbf{w}_4 = 41$) results in a scaled version of the CENS features simulating a tempo change of $10/13 \approx 0.77$. The eight pairs cover global tempo variations of $\pm 40\%$. For each of the resulting query sequences, we separately calculate a distance function Δ_j, $j \in [1 : 8]$. Finally, the joint distance function $\Delta_{\min} : [0 : M-1] \to [0,1]$ is defined by $\Delta_{\min}(i) := \min(\Delta_1(i), \ldots, \Delta_8(i))$. Then the parameter $i_0 \in [0 : M-1]$ minimizing $\Delta_{\min}$ yields the best match between the query and a database subsequence starting with w_{i_0}. To obtain the length of this subsequence, let $j_0 \in [1 : 8]$ denote the index that minimizes $\Delta_j(i_0)$. Then the best match is given by (i_0, N_{j_0}) corresponding to the subsequence $(w_{i_0}, \ldots, w_{i_0+N_{j_0}-1})$. To obtain further matches, one can proceed as described above.

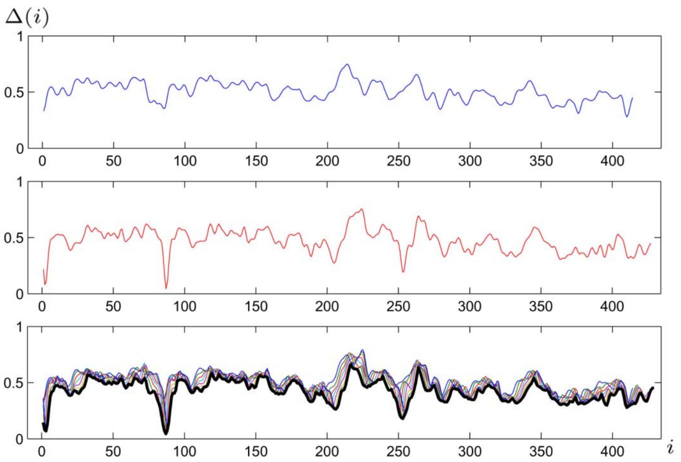


Fig. 2. Query Q consists of the first 22 s of a Bernstein interpretation and the database D of a Karajan interpretation (443 s) of the first movement of Beethoven's Fifth. The tempo in the Bernstein interpretation is much slower (roughly 80%) than the one in the Karajan interpretation. Top: Distance function Δ_4 for $\mathrm{CENS}^{41}_{10}[Q]$. Middle: Distance function Δ_7 for $\mathrm{CENS}^{53}_{13}[Q]$. Bottom: Distance functions Δ_j for $j \in [1 : 8]$ and joint distance function $\Delta_{\min}$ (bold line).

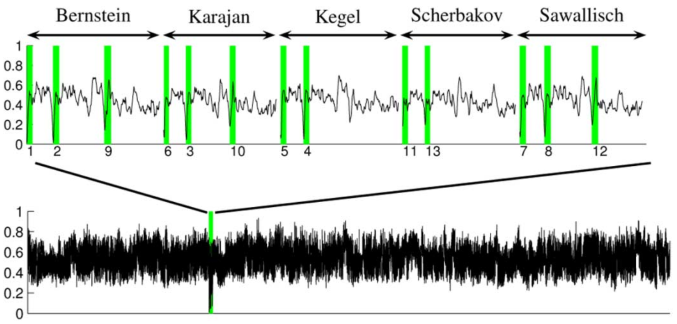


Fig. 3. Bottom: Distance function $\Delta_{\min}$ for the entire database (112 h) w.r.t. the Bernstein query (21 s) as specified in Fig. 2. Note that most of the correct matches ($\Delta_{\min}$ is close to zero) are clearly separated from the remaining pieces ($\Delta_{\min}$ is above the value 0.2). Looking for all matches with $\Delta_{\min}$-costs below the threshold 0.2 results in 13 matches, which are among the 15 "correct" matches. Top: Enlargement showing the five interpretations of the first movement of Beethoven's Fifth and the 13 top matches.

A similar strategy can be employed to account for a global difference in the musical key between the query and the database matches. The basic idea is to simulate all possible twelve modulations by cyclically shifting the 12-dimensional CENS vectors of the query sequence, see [25]. The resulting twelve query versions are then processed as in the case of the global tempo simulation. Note that as both the simulation of different modulations and the above tempo changes are performed for the query signal only, this step does not involve an increase of the database size.

We have evaluated this matching strategy on two databases, one mainly consisting of classical music and the other containing different genre of popular music (see Section V-A for details). Altogether, both databases contain a total of 152 h of audio recordings (112 h of classical and 40 h of popular music).

The classical database includes five different interpretations of Beethoven's Fifth—four orchestral versions conducted by Bernstein, Karajan, Kegel, and Sawallisch, respectively, and Liszt's piano transcription played by Scherbakov. Let Q denote the query consisting of the first 21 measures (22 s) of Bernstein's interpretation. Note that the first 21 measures of this symphony appear once more in the repetition of the exposition and, with some slight modifications, in the recapitulation. Therefore, altogether, we expect 15 "correct" matches that are considered similar to the query. Using our matching procedure, we automatically determined the best 15 matches in the entire database w.r.t. $\Delta_{\min}$. Those 15 matches contained 14 of the 15 "correct" occurences—only the 14th match (having distance 0.217), which corresponds to some excerpt of Schumann's third symphony was "wrong," i.e., a false positive match. Furthermore, it turned out that the first 13 matches are exactly the ones having a $\Delta_{\min}$-distance below the value 0.2 from the query, see also Fig. 3. Note that even the occurrences in the exposition of Scherbakov's piano version were correctly identified as 11th and 13th match, even though differing significantly in timbre and articulation from the orchestral query.

Regarding the popular music database, our audio matching approach was generally successful in retrieving different interpretations of query excerpts corresponding to demo versions, live recordings, or even severely distorted bootlegs. As in the case of classical music, the matching algorithm works best for passages exhibiting salient harmonic progressions as they mostly occurred in our queries taken from pop, folk, and rock music. In a considerable number of cases, there are stereotypic repetitions of the same harmonic progression within the same song (such as in the song Clocks by the artist Coldplay) leading to a high number of matches within the same piece. As could be expected from the particular choice of features, our matching approach is problematic in case of music which is extremely dominated by rhythmic components without exhibiting salient harmonics, such as hard rock or metal.

For running times of this matching procedure and the index-based matching procedure to be described, we refer to Section V.

III. Codebook Selection for CENS Features

The matching procedure described so far has a running time that linearly depends on the length of the database. In order to speed up the matching procedure, we will introduce an index-based approach using an inverted file index in Section IV. This kind of indexing depends on a suitable *codebook* that consists of a finite set of characteristic CENS vectors. In this section, we discuss various strategies for constructing such codebooks.

Recall that the set $\mathcal{F}$ of possible CENS features consists of all points on the unit sphere S^{11} with nonnegative entries, see (1). Then a codebook is given by a finite set $\mathcal{C}_R = \{c_1, \ldots, c_R\} \subset \mathcal{F}$, where each index $r \in [1:R]$ corresponds to a feature class determined by a nearest neighbor criterion. More precisely, an arbitrary feature vector $v \in \mathcal{F}$ is assigned to the class label $\mathcal{Q}[v] \in [1:R]$ defined by

$$\mathcal{Q}[v] := \arg\min_{r\in[1:R]} \arccos(\langle v, c_r\rangle). \tag{4}$$

(In cases where the minimizing index is not unique, we randomly choose one of these indices.) In other words, the vector $c_{\mathcal{Q}[v]}$ minimizes the angle to v among all elements in the $\mathcal{C}_R$. The function $\mathcal{Q} : \mathcal{F} \to [1:R]$ is also referred to as *quantization function* associated with the codebook $\mathcal{C}_R$. In the remainder of this section, we describe two fundamentally different approaches for selecting the codebook $\mathcal{C}_R$.

A. Codebook Selection by Unsupervised Learning

A common strategy of selecting a codebook without exploiting any further knowledge about the underlying data is based on unsupervised clustering. In the following, we show how the well-known LBG (Linde–Buzo–Gray) algorithm [32] for vector quantization in the Euclidean space can be adapted to perform clustering in the spherical feature space $\mathcal{F} \subset S^{11}$.

Let $W = (w_0, \ldots, w_{M-1})$ denote the database feature sequence, which will be used for constructing a codebook $\mathcal{C}_R = \{c_1, \ldots, c_R\} \subset \mathcal{F}$. The LBG algorithm consists of two steps. In an *initialization* step, an initial codebook $\mathcal{C}_R^0 = \{c_1^0, \ldots, c_R^0\} \subset \mathcal{F}$ is chosen. In this work, we randomly chose this initial codebook from a uniform distribution on $\mathcal{F}$ (it may also be randomly chosen from V). The parameter R is chosen to yield a suitable number of inverted lists, see Section IV. Using the nearest neighbor assignment on the sphere, the codebook induces a quantization function $\mathcal{Q}^0$ analogous to (4). The mean angular *quantization error* is then defined by

$$\varepsilon^0 := \frac{1}{M} \sum_{m=0}^{M-1} \arccos\left(\left\langle w_m, c^0_{\mathcal{Q}^0[w_m]}\right\rangle\right). \tag{5}$$

In the subsequent *iterative* step, the following procedure of alternately selecting a new codebook and updating the quantization function is repeated for steps $\ell = 1, 2, \ldots$ until a suitable stop criterion is satisfied.

1) Let $P_r := (\mathcal{Q}^{\ell-1})^{-1}(r)$ denote the set of features which are assigned the class label $r \in [1:R]$ by the quantization function $\mathcal{Q}^{\ell-1}$. Define the new codebook $\mathcal{C}_R^\ell = \{c_1^\ell, \ldots, c_R^\ell\}$ by updating the codebook vectors according to

$$c_r^\ell := \pi^{\mathcal{F}}\left(\frac{1}{|P_r|}\sum_{v\in P_r} v\right) \tag{6}$$

for all $r \in [1:R]$.

2) Determine a new quantization function $\mathcal{Q}^\ell : \mathcal{F} \to \mathcal{C}_R^\ell$ by setting

$$\mathcal{Q}^\ell(w_m) := \arg\min_{r\in[1:R]} \arccos\left(\left\langle w_m, c_r^\ell\right\rangle\right). \tag{7}$$

3) Calculate the new mean angular quantization error

$$\varepsilon^\ell := \frac{1}{M} \sum_{n=0}^{M-1} \arccos\left(\left\langle w_m, c^\ell_{\mathcal{Q}^\ell[w_m]}\right\rangle\right). \tag{8}$$

The iteration is terminated as soon as the difference $|\varepsilon^\ell - \varepsilon^{\ell-1}|$ falls below a suitable threshold. We then define $\mathcal{Q} := \mathcal{Q}^\ell$ and $\mathcal{C}_R := \mathcal{C}_R^\ell$.

Besides choosing a spherical distance (angles) on $\mathcal{F}$, the difference to the classical LBG-approach is the additional projection on the unit sphere expressed by the operator $\pi^{\mathcal{F}}$. This projection is necessary because the convex combination $|P_r|^{-1}\sum_{v\in P_r} v$ computed in (6) is generally not located on $\mathcal{F}$. Instead, one could use spherical averages to better account for the spherical geometry of $\mathcal{F}$, see [33]. In practice, however, the effects on the final clustering result are only marginal. The convergence of the modified LBG-algorithm is somewhat slower than in the classical case. However, as the codebook is computed offline, our proposed algorithm turns out to be sufficiently fast for our purpose. As an example, the algorithm required $\ell = 127$ iterations to select a codebook of size $R = 200$ when choosing the test database DB55 (which is introduced in Section V-A and has a size of $M = 186{,}929$ vectors) as training set W.

B. Codebook Selection Based on Musical Knowledge

One main advantage in using the unsupervised learning procedure for codebook selection is the freedom in choosing the number R of codebook vectors, which is an important parameter for the efficiency of index-based matching, see Section IV. However, the learning approach depends on the availability of a suitable set of training data to yield a codebook that generalizes to an arbitrary data set. To overcome this limitation, we directly construct a codebook without resorting to any training data by exploiting musical knowledge.

The main idea is that a CENS vector contains some explicit information regarding the harmonic content of the underlying audio fragment. For example, a CENS vector close to

$$\frac{1}{\sqrt{3}}(1,0,0,0,1,0,0,1,0,0,0,0) \in \mathcal{F} \tag{9}$$

indicates the harmonic relation to C major. Since our database mainly consists of harmonic Western music based on the equal-tempered scale, one can expect clusters of CENS vectors that correspond to single notes, intervals (combination of two notes), or certain triads (combination of three notes) such as major, minor, augmented, or diminished chords. An analysis of 55 h of audio recordings chosen from our test database shows that, for more than 95% of all extracted CENS vectors, more than 50% of a vector's energy is contained in at most four of the twelve components. In other words, for most CENS vectors, the energy is concentrated in only a few components, which is the motivation for the following procedure. Let $\delta_1,\ldots,\delta_{12} \in \mathcal{F}$ denote the 12 unit vectors, which correspond to pure chroma that contain the entire energy within one component. Let $C_i \subset \mathcal{F}$, $i \in [1:12]$, denote the set of vectors with i dominant components

$$C_i := \left\{\pi^{\mathcal{F}}(\delta_{k_1}+\cdots+\delta_{k_i}) \mid 1 \le k_1 < \cdots < k_i \le 12\right\}. \tag{10}$$

As basic building blocks for our codebook, we will consider all vectors with up to four dominant components resulting in

$$|C_1|+|C_2|+|C_3|+|C_4| = \sum_{i=1}^{4}\binom{12}{i} = 793 \tag{11}$$

vectors. Note that several of these vectors such as $\pi^{\mathcal{F}}(1,1,1,1,0,0,0,0,0,0,0,0)$ may correspond to harmonically rare combinations of notes. However, due to the temporal blurring of the CENS vectors, such combinations may occur particularly in passages such as harmonic transitions, chromatic scales, or ornamentations. Since the number of resulting vectors is well under control, we do not further thin out the set of building blocks.

To account for the harmonics contained in the acoustic realization of a single musical note, we introduce an additional note model. As an example, let us consider the concert pitch A4 of fundamental frequency $f = 440$ Hz, which corresponds to the MIDI pitch $p = 69$ and has the chroma index $(p \bmod 12)+1 = 10$. Then, an acoustic realization of A4 basically consists of a weighted mix of harmonics having the frequencies $h \cdot f$ with $h \in \mathbb{N}$. Generally, let p denote the MIDI pitch of some musical note with fundamental frequency $f = 2^{(p-69)/12}\cdot 440$ Hz and let $\gamma(p,h)$ denote the chroma index of the hth harmonic of p with respect to the equal-tempered scale. It is easy to show that

$$\gamma(p,h) = ((p + \operatorname{round}(12\log_2 h)) \bmod 12) + 1. \tag{12}$$

For example, considering the first eight harmonics, the chroma indices $\gamma(p,h)$ for $h = 1,2,\ldots,8$ with respect to the musical note C4 (MIDI pitch $p = 60$) are given by 1, 1, 8, 1, 5, 8, 11, 1, which correspond to the chroma C, C, G, C, E, G, B, C. In our note model, we consider eight harmonics, where the harmonic components are weighted according to a weight vector $(\alpha_1,\ldots,\alpha_8)$. In our experiments, we set $\alpha_h = 1$ for $h = 1, 2, 3$ and $\alpha_h := 1/(h-2)$ for $h = 4,\ldots,8$. Actually, as our experiments showed, the particular number of harmonics has only a marginal effect on the overall matching result. The reason we chose eight harmonics is that these harmonics still can be reasonably assigned to pitches in the equal-tempered scale. Based on the latter, the unit vector δ_k, which represents all pitches p of chroma index $k = (p \bmod 12)+1$, is replaced by the linear combination

$$\tilde{\delta}_k := \pi^{\mathcal{F}}\left(\sum_{h=1}^{8}\alpha_h\cdot\delta_{\gamma(p,h)}\right) \tag{13}$$

for a p with chroma index k. Note that $\tilde{\delta}_k$ can be obtained from $\tilde{\delta}_1$ by cyclically shifting the components of $\tilde{\delta}_1$ by $k-1$ positions.

Having incorporated harmonics, we now replace the sets C_i in (10) by modified versions

$$\tilde{C}_i := \left\{\pi^{\mathcal{F}}(\tilde{\delta}_{k_1}+\cdots+\tilde{\delta}_{k_i}) \mid 1 \le k_1 < \cdots < k_i \le 12\right\}. \tag{14}$$

The final codebook is then defined by $\mathcal{C}_{793} := \tilde{C}_1 \cup \tilde{C}_2 \cup \tilde{C}_3 \cup \tilde{C}_4 \subset \mathcal{F}$.

We conclude this section by noting that in our experiments both of the above approaches to codebook selection turn out to possess individual benefits as will be discussed in Section V.

IV. Index-Based Audio Matching

We now describe how our audio matching procedure of Section II can be supplemented by a much faster index-based matching procedure that employs an inverted file index [11], [28]. The main idea is to classify (quantize) the database CENS features with respect to a selected codebook $\mathcal{C}_R$ and then to index the features according to their class labels $r \in [1 : R]$. This results in an inverted list (also referred to as inverted file) for each label. Exact audio matching can then be performed efficiently by intersecting suitable inverted lists (Section IV-A). To account for musical variations, we soften the notion of exact matching by introducing various fault tolerance and ranking mechanisms (Section IV-B). To further improve the ranking, we finally combine the concepts of index-based and diagonal matching, which yields a ranked list of audio matches to the original query (Section IV-C).

A. Exact Matches

In this section, we introduce the concept of exact matches (with respect to a given quantization function) and show how these matches can be computed efficiently. Let $W = (w_0, \ldots, w_{M-1})$ be the database CENS sequence of the database D. We fix a codebook $\mathcal{C}_R = \{c_1, \ldots, c_R\} \subset \mathcal{F}$ and denote its associated quantization function by $\mathcal{Q}$, see (4). Furthermore, let $r_m := \mathcal{Q}[w_m]$, $m \in [0 : M-1]$, denote the quantized feature vectors and

$$\mathcal{Q}[W] := (r_0, r_1, \ldots, r_{M-1}) \qquad (15)$$

the quantized database sequence. For each class label $r \in [1 : R]$, we then create an inverted list $L(r)$, which consists of all index positions m such that the vector w_m is quantized to class r:

$$L(r) := \{m \in [0 : M-1] \mid r_m = r\}. \qquad (16)$$

The inverted file index of the database, which can be computed in a preprocessing step, consists of all inverted lists $(L(r))_{r \in [1:R]}$.

To process a query Q, it is converted into a query CENS sequence $V = (v_0, v_1, \ldots, v_{N-1})$ (with respect to suitable parameters $\mathbf{w}$ and $\mathbf{d}$). Next, the query sequence is quantized to yield

$$\mathcal{Q}[V] := (s_0, s_1, \ldots, s_{N-1}) \qquad (17)$$

where we set $s_n := \mathcal{Q}[v_n]$ for $n \in [0 : N-1]$. Then, an *exact match* is defined to be a tuple (k, N) such that $\mathcal{Q}[V]$ is a subsequence of consecutive feature vectors in $\mathcal{Q}[W]$ starting from index k. In other words, writing in this case $\mathcal{Q}[V] \sqsubset_k \mathcal{Q}[W]$, one obtains

$$\mathcal{Q}[V] \sqsubset_k \mathcal{Q}[W] :\Leftrightarrow \forall\, n \in [0 : N-1] :\ s_n = r_{k+n}. \qquad (18)$$

The set of all match positions yielding exact matches of length N is given by

$$H(\mathcal{Q}[V]) := \{k \in [0 : M-1] \mid \mathcal{Q}[V] \sqsubset_k \mathcal{Q}[W]\}. \qquad (19)$$

Now, it is not hard to see that this set can be calculated by intersecting suitably shifted inverted lists:

$$H(\mathcal{Q}[V]) = \bigcap_{n \in [0:N-1]} (L(s_n) - n) \qquad (20)$$

where the difference of an inverted list and a natural number is defined elementwise. Equation (20) can be proven as follows: There is a match at position k, iff each of the $n \in [0 : N-1]$ query features s_n occurs at position $k+n$ in the database sequence; hence, $k+n \in L(s_n)$ must hold. The latter is equivalent to $k \in (L(s_n) - n)$ for all n, which proves our claim. As each inverted list can be stored as a *sorted* list of integers, the intersections in (20) can be performed very efficiently by linearly processing sorted integer lists, see [11].

B. Fault Tolerance Mechanisms

Even though the usage of CENS features as well as the quantization step introduce a high degree of robustness to variations in dynamics, instrumentation, and articulation, the requirement of exact matching on the quantized feature level is still too restrictive in order to obtain most of the relevant matches. In the following, we introduce several fault tolerance mechanisms to soften the strict requirements of exact matching. In contrast to existing generic approaches for fuzzy or approximate string matching (e.g., online methods based on the edit distance [34] or index-based approaches based on suffix trees [35]) our fuzzy concepts are adapted to the matching scenario addressed in this paper. In particular, the proposed mechanisms take into account the topology of the underlying feature space (e.g., for generating the fuzzy query sequences introduced in this section) and are compatible for use with our indexing strategy based on inverted lists.

1) Fuzzy Matches: The first general mechanism is known as *fuzzy search*, see [11]. In the audio matching context, we use this mechanism to cope with the problem that arises when slight differences in the extracted CENS features lead to different assignments of codebook vectors in the quantization step (4). The idea is to admit, for each position of the query sequence, a whole set of possible, alternative class labels instead of a single one. To this end, we introduce the concept of a *fuzzy quantization function* $\mathcal{Q}_\lambda^\varrho$ for some $\lambda \in \mathbb{N}$ and $\varrho \in \mathbb{R}_{>0}$. Instead of using only the nearest codebook vector $\mathcal{Q}[v]$ for quantizing a CENS feature $v \in \mathcal{F}$, we now assign an entire set $\mathcal{Q}_\lambda^\varrho[v] \subset \mathcal{F}$ of alternative codebook vectors. By definition, this set contains at least the nearest neighbor $\mathcal{Q}[v]$ and all of those additional $\lambda - 1$ next nearest neighbors that have an angular distance to v below the threshold ϱ. Here, the threshold condition is introduced to avoid outliers in the assignment. In our experiments, $3 \leq \lambda \leq 7$ and $\varrho = 0.15\pi$ turn out to be reasonable parameter values. Now, the query sequence V is quantized with respect to $\mathcal{Q}_\lambda^\varrho$ to yield a sequence

$$\mathcal{Q}_\lambda^\varrho[V] := (S_0, S_1, \ldots, S_{N-1}) \qquad (21)$$

where we set $S_n := \mathcal{Q}_\lambda^\varrho[v_n]$ for $n \in [0 : N-1]$. Extending the definition in (18), a *fuzzy match* is a tuple (k, N) such that $\mathcal{Q}_\lambda^\varrho[V] \sqsubset_k \mathcal{Q}[W]$, where

$$\mathcal{Q}_\lambda^\varrho[V] \sqsubset_k \mathcal{Q}[W] :\Leftrightarrow \forall\, n \in [0 : N-1] :\ S_n \ni r_{k+n}. \qquad (22)$$

The set of all match positions yielding fuzzy matches of length N is given by

$$H\left(\mathcal{Q}_\lambda^\varrho[V]\right) := \left\{k \in [0 : M-1] \mid \mathcal{Q}_\lambda^\varrho[V] \sqsubset_k \mathcal{Q}[W]\right\}. \qquad (23)$$

Finally, defining the lists $L(S_n) := \bigcup_{s \in S_n} L(s)$, one obtains

$$H\left(\mathcal{Q}_\lambda^\varrho[V]\right) = \bigcap_{n \in [0:N-1]} \left(L(S_n) - n\right) \tag{24}$$

thus yielding an efficient algorithm to compute all fuzzy matches from the inverted file index. Note that if $S_n = \{s_n\}$ for all $n \in [0 : N-1]$ one obtains the case of exact matches.

2) Fuzzy Matches With Mismatches: As a second general fault tolerance mechanism, we adopt the concept of *mismatch search*, see [11]. Here, the idea is to admit a certain number of mismatches when matching the (fuzzy) query sequence with a database subsequence. More precisely, given the fuzzy query $\mathcal{Q}_\lambda^\varrho[V] = (S_0, S_1, \ldots, S_{N-1})$, we define a function $\mu : [0 : M-1] \to [0 : N]$ that counts the following multiplicities:

$$\mu(m) := \left|\{n \in [0 : N-1] \mid m \in (L(S_n) - n)\}\right|. \tag{25}$$

Obviously, $\mu(k) = N$ if and only if $k \in H\left(\mathcal{Q}_\lambda^\varrho[V]\right)$, cf. (24). Generalizing the concept of a fuzzy match, we call a pair (k, N) a *fuzzy match with ν mismatches* for some mismatch parameter $\nu \in [0 : N]$ if $\mu(k) = N - \nu$. In this case, the condition $r_{k+n} \in S_n$ for $n \in [0 : N-1]$ as given in (22) is violated at exactly ν positions. The mismatch function can be computed efficiently either by dynamic programming [11] or by hashing techniques [36].

3) Multiple Queries: The third mechanism, which accounts for global tempo variations, proceeds in analogy to the one described in Section II. Instead of only using a single query feature sequence $V = \mathrm{CENS}_{10}^{41}[Q]$, we use eight different query sequences $\mathrm{CENS}_{\mathbf{d}_j}^{\mathbf{w}_j}$, $1 \leq j \leq 8$, which are then processed separately. Similarly, one can account for global variations in the musical key by processing the query sequences obtained by cyclically shifting the components of the CENS vectors.

C. Retrieval Scenario and Ranking

From an information retrieval point of view, our index-based approach to audio matching basically consists of two stages, which are illustrated in Fig. 4. In the *indexing stage*, the music database D is converted into the feature sequence $\mathrm{CENS}_{10}^{41}[D]$, which is then quantized with respect to some fixed codebook $\mathcal{C}_R$ and indexed via inverted lists, see Fig. 4(a). The resulting inverted file index is independent of the query, the fault tolerance stettings, and the ranking strategies described at the end of this section. In the *query and retrieval stage*, the user supplies a query Q in form of some audio clip, which typically has a duration between 10 and 30 s. Optionally, the user may specify certain fault tolerance parameters that affect the fuzzyness of the query, the number of admissible mismatches, or the tolerance to global variations in tempo and musical key. Based on these parameters and the codebook $\mathcal{C}_R$, the query Q is transformed into multiple fuzzy queries which are then processed by means of the inverted file index. Finally, after some suitable postprocessing, the system returns a ranked list of matches, see Fig. 4(b).

In the remaining part of this section, we describe two different ranking strategies used in the postprocessing step. The first strategy ranks the matches according to the number of mismatches, which can be derived from the multiplicity function μ defined in (25). Let (k, N) denote a retrieved match with multiplicity $\mu(k) \in [0 : N]$. Note that due to the usage of different query sequences $\mathrm{CENS}_{\mathbf{d}_j}^{\mathbf{w}_j}$ (having a length of N_j, see Section II-B), the length of the retrieved matches may vary. To compensate for the respective length, we define the ranking value $\Theta(k, N)$ of the match (k, N) to be the quotient

$$\Theta(k, N) := \frac{\mu(k)}{N} \in [0, 1]. \tag{26}$$

In case of an exact match or a fuzzy match without mismatches, one obtains $\Theta(k, N) = 1$. The quality of matches decreases with a decreasing Θ-value. In order to avoid an excessive number of matches in the retrieval process, one can either introduce a fixed upper bound for the number of matches to be retrieved or one can *a priori* restrict the matches by introducing a lower bound for the ranking value (which can also be exploited to further speed up the computation of μ).

The ranking function Θ is relatively coarse and assumes only a small number of different values in particular for short queries. Furthermore, a codebook $\mathcal{C}_R$ with a relatively small number of elements as well as relatively large parameters λ and ϱ in the fuzzy quantization function $\mathcal{Q}_\lambda^\varrho$ may result in a large number of false positives and unexpected matches with a high or even maximal Θ-rank. To improve the ranking as well as to eliminate large overlaps in the retrieved matches, we further postprocess the top matches by the diagonal matching procedure described in Section II-B. For each of those matches, we consider the corresponding subsequence in the database sequence W. To obtain a more flexible diagonal matching, each such subsequence is extended by a small number of vectors to the left and the right. Finally, diagonal matching is performed on each of the extended subsequences. The resulting matches are ranked according to the values assumed by the joint distance function $\Delta_{\min}$, see Section II-B. To avoid multiple matches at very close positions within the same song, a match candidate is rejected if it overlaps a previously found (and hence higher ranking) match by more than 30%.

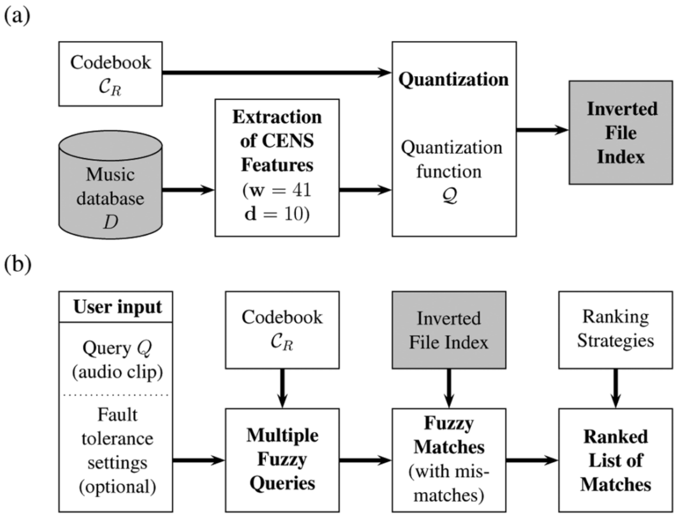


Fig. 4. Overview of our index-based audio matching procedure. (a) The indexing stage. (b) The query and retrieval stage.

In a sense, the index-based matching can be seen as an efficient way to significantly reduce the search space (database) to a small fraction (top matches obtained by index-based audio matching), while retaining a superset of the desired matches. Then the more cost-intensive diagonal matching is performed only on a small fraction of the database.

V. Experimental Results

We have conducted various experiments to evaluate our index-based audio matching procedure. In this section, we first describe the music collection used in our evaluation (Section V-A). Prior to investigating efficiency issues, we qualitatively compare the results obtained from the pure diagonal matching procedure and the proposed index-based method (Section V-B). A special focus is put on comparing the audio matching results with respect to various codebooks constructed by means of the learning-based and knowledge-based approaches (Section V-C). Finally, as a main result of this paper we report on the speed-up factors obtained by our index-based matching approach (Section V-D).

A. Test Collections

For our experiments, we set up two test databases mainly consisting of Western music. Our first database (DB112) contains 112 h of classical music, requiring 16.5 GB of disk space (uncompressed mono, 22 050 Hz). The database comprises 1167 audio files reflecting a wide range of classical music, including, among others, pieces by Bach, Bartok, Bernstein, Beethoven, Chopin, Dvorak, Elgar, Mozart, Orff, Ravel, Schubert, Shostakovich, Vivaldi, and Wagner (about 30 composers in total). In particular, it contains all Beethoven symphonies, all Beethoven piano sonatas, all Mozart piano concertos, several Schubert and Dvorak symphonies—many of the pieces in several versions. In most cases, the audio files constitute individual *movements* of musical works. On the movement level, DB112 contains about 500 different musical pieces. Some of the orchestral pieces are also contained as piano arrangements or synthesized MIDI-versions (about 75 synthesized recordings are contained, where the specific synthesizer is not of significant relevance with regard to the retrieval results). A subset of this database corresponding to 55 h of music (this database will be called DB55) was used for training the codebook in the unsupervised learning approach described in Section III-A. The second database (DB40) consists of about 40 h of popular music, mostly taken from the Rock, Hardrock, Folk, and general Pop genres. DB40 includes about 500 recordings of about 400 different works. For 60 works, there are more than one recording, the different recordings ranging from remixes and demos to official recordings of live performances and bootlegs. In addition to containing different types of musical variations, pieces within the latter two categories are sometimes severely distorted. For the cases where different versions of a piece of music are contained in DB40, we included all pieces from at least one CD containing the respective piece of music, at least another album of the particular artist, and at least another album from an artist with pieces of a similar musical style. This allows us to test the sensitivity of our matching procedure with respect to various types of similarity regarding style, genre, and instrumentation.

To qualitatively evaluate the proposed audio matching methods, we created a ground truth data set of relevant query results for 36 queries (Q-GT36), which were manually selected from DB112. All of those queries constitute popular excerpts of classical pieces (such as the formerly introduced excerpt of Beethoven's Fifth Symphony, the theme of Ravel's Bolero, or different excerpts of Bach's Goldberg Variations) having durations between 10 and 40 s. Furthermore, for our quantitative tests, we generated three sets (Q-R10, Q-R15, Q-R20) each consisting of roughly 1000 random queries having durations of 10, 15, and 20 s, respectively. The particular choice of 10, 15, and 20 s was made to evaluate the system's efficiency for gradually changing query lengths.

B. Diagonal Versus Index-Based Matching

For an informal evaluation and a discussion of the pure diagonal matching approach, we refer to [4]. We now summarize some of the results of our systematic experiments, which have been conducted for both the diagonal and the index-based audio matching procedure. In what follows, the index-based matching approach is always postprocessed using both of the ranking strategies introduced in Section IV-C.

In a first experiment, we investigated the quality of the retrieval results for various combinations of audio matching parameters using queries from Q-GT36. A main goal of this experiment was to evaluate suitable parameter settings for the subsequent tests. In particular, we tested three different codebooks $\mathcal{C}_R$ with sizes $R \in \{50,100,200\}$ obtained from the unsupervised learning procedure, five different parameters $\lambda \in \{3,4,5,6,7\}$ with a threshold of $\varrho = 0.15\pi$ used in the fuzzy quantization function $\mathcal{Q}_\lambda^\varrho$, as well as four different lower bounds $\theta \in \{0.2, 0.3, 0.4, 0.5\}$ for the Θ-ranking value.

As a first result, it turns out that the codebook $\mathcal{C}_{200}$ outperforms the smaller codebooks $\mathcal{C}_{50}$ and $\mathcal{C}_{100}$. A manual inspection of the retrieval results indicates that the number of vectors in the smaller codebooks is too low to sufficiently discriminate true and false positive matches. In particular, when using the codebook $\mathcal{C}_{100}$ the number of match candidates with a Θ-value greater than 0.3 (less than 70% mismatches) generally exceeds the corresponding number for $\mathcal{C}_{200}$ by almost one order of magnitude. As a further consequence of smaller codebooks, the inverted lists get longer which in turn results in increased running times ranging from 10% to 20% in the latter case (with fixed values of λ and θ). To find suitable parameter settings for λ and θ, we started with $\mathcal{C}_{200}$ and the query set Q-GT36, gradually increasing λ until the overall number of false negatives did not further increase. To compensate for the simultaneously increasing number of match candidates, the threshold θ of the required minimum rank was suitably adapted. In this process, the parameter combination $\lambda = 5$ (at most five alternatives in each fuzzy set) and $\theta = 0.4$ (at least 40% of the features have to match) turned out to yield the best retrieval results.

In a second experiment, we qualitatively compared the index-based matching approach (codebook size $R = 200$, fuzzy parameters $\lambda = 5$ and $\theta = 0.4$) and the pure diagonal matching approach based on the queries from Q-GT36. Using the manually annotated ground truth, we obtain precision-recall (PR) values from the top ten matches for each of the two methods. Fig. 5

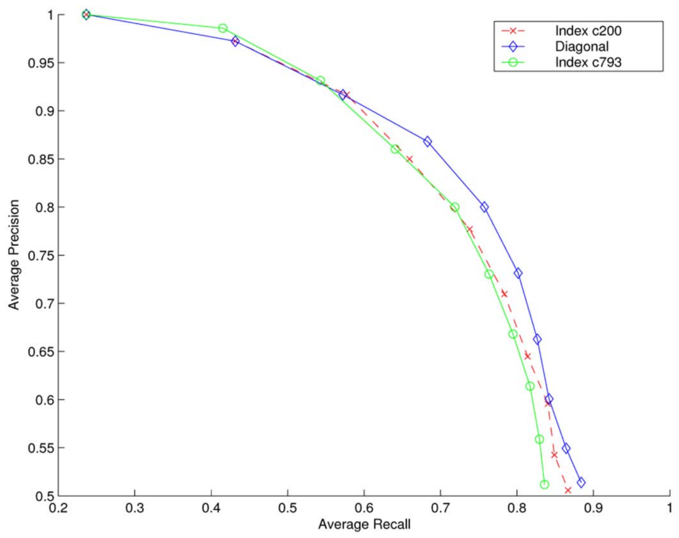


Fig. 5. Average precision-recall (PR) values on the query set Q-GT36 for the top ten retrieval results. Diamond curve: PR diagram for pure diagonal matching. Cross curve: PR diagram for index-based matching (learning-based codebook $\mathcal{C}_{200}$, $\lambda = 5, \theta = 0.4$). Circle curve: PR diagram for index-based matching (knowledge-based codebook $\mathcal{C}_{793}$, $\lambda = 7, \theta = 0.3$).

TABLE I
COMPARISON OF AUDIO IDENTIFICATION CAPABILITIES OF DIAGONAL AND INDEX-BASED AUDIO MATCHING. THE COLUMNS INDICATE THE AVERAGE RECALL VALUES FOR THE TOP MATCH, THE TOP TWO, AND THE TOP THREE MATCHES, RESPECTIVELY

		Average Recall		
		@1	@2	@3
Q-R10	Diagonal	0.97	1	1
	Index	0.95	1	1
Q-R15	Diagonal	0.972	0.998	0.999
	Index	0.957	0.996	0.999
Q-R20	Diagonal	0.972	0.999	0.999
	Index	0.965	0.999	1

shows the corresponding PR-diagram obtained from averaging the PR values over all 36 queries. While diagonal matching appears to be slightly better in the range of the top four to five matches, the PR values for the top three matches and the top seven to ten matches almost coincide. Note that because there are only about six correct matches on average for a query in Q-GT36, there is a certain bias in the PR-curves, which consider the top ten matches. However, this bias does not affect the comparability of the curves relative to each other.

In a third experiment, we tested how the two audio matching procedures perform in the classical *audio identification* scenario, where the objective is to identify only the query within the audio database (rather than identifying all musically similar audio clips). To this end, we used the 3000 queries from Q-R10, Q-R15, and Q-R20, which were randomly drawn from DB112, as an input. For both matching procedures, Table I shows the average recall values for the top match, the top two, and the top three matches, respectively. Considering only the top match, the percentage of correct identifications increases with the query length. Considering the top two matches, both procedures yield almost optimal identification results. Here, note that the identification task is not a trivial one: due to shifts in the analysis windows used in computation of the chroma and CENS features (e.g., using CENS^{41}_{10}-features which have a temporal resolution of 1 Hz, these shifts can be up to half a second, see Section II-A), the CENS feature sequence V of the query generally does not coincide with the corresponding CENS feature subsequence within the database sequence W. Most of the confusion in the audio identification are due to more or less identical repetitions of the query within the respective music piece.

TABLE II
COMPARISON OF AUDIO IDENTIFICATION CAPABILITIES OF DIAGONAL AND INDEX-BASED AUDIO MATCHING FOR DISTORTED QUERY SIGNALS (TEST SET Q-R15)

		Average Recall		
		@1	@2	@3
Lame MP3-compression @65kbps	Diagonal	0.968	0.998	0.999
	Index	0.956	0.999	1
Time stretching (100% to 150% tempo)	Diagonal	0.962	0.998	0.999
	Index	0.944	0.998	1
Strong 3D echo-chamber effect	Diagonal	0.925	0.979	0.985
	Index	0.881	0.954	0.966

Further experiments show that the audio identification still works with high precision even in the case that the queries are severely distorted. Table II shows the corresponding average recall values for three different types of transformations performed on all of the queries from Q-R15. The first transformation is lossy audio compression using the Lame MP3 audio encoder at the low bitrate of 65 kb/s. As the compression follows perceptual criteria, the signals' harmonic content is retained, resulting in recall values which almost correspond to those obtained for the original signals. More interesting are the relatively high recall values for the second transformation, being a nonlinear temporal deformation (time stretch) where the tempo of each query is linearly increased from the original tempo to the 1.5-fold tempo. To simulate an acoustic signal transmission, the third transformation consists of applying a strong 3-D echo-effect (*Ambient Metal Room* preset from the CoolEdit 2000 softwares' delay effects) to the query signals. Due to the echo effect, the harmonies are smeared over an interval of a few seconds, which significantly affects the signal's energy distribution in the chroma bands. However, even in this case, the average recall for the top three matches is still high.

C. Comparison of Codebooks

In Section III, we have introduced both a learning-based and a knowledge-based strategy for codebook selection. Various experiments have been conducted to investigate the effect of the respective codebook on the retrieval process. Similar to our second experiment, we have qualitatively compared the index-based matching approach for the codebook $\mathcal{C}_{200}$ (learning-based) with $\lambda = 5$ and $\theta = 0.4$ and the codebook $\mathcal{C}_{793}$ (knowledge-based) with $\lambda = 7$ and $\theta = 0.3$. Here, the fuzzy and ranking parameters were in each case chosen to yield optimum retrieval results within the above experimental setting. Note that due to the larger number of codebook vectors in $\mathcal{C}_{793}$, more relaxed parameters λ and θ are chosen. Fig. 5 shows the average PR diagrams over the query set Q-GT36 using the two different codebooks. While the retrieval results for our set of queries are almost eqivalent, the manually constructed codebook yields slightly better results for the top three matches. In this case, the index-based matching even outperforms diagonal matching.

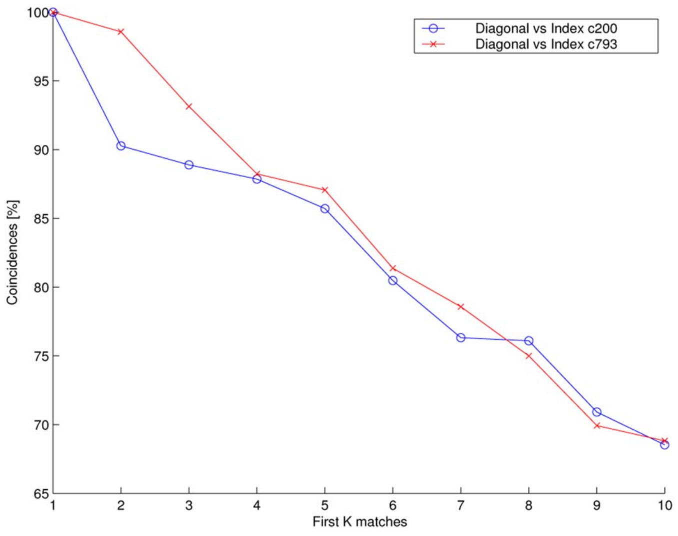


Fig. 6. Average coincidence of top K matches in percent: Circle curve: diagonal matching compared to index using the learning-based codebook $\mathcal{C}_{200}$ ($\lambda = 5, \theta = 0.4$). Cross curve: diagonal matching compared to index using the knowledge-based codebook $\mathcal{C}_{793}$. ($\lambda = 7, \theta = 0.3$).

We furthermore directly compared the retrieval results of the two index-based approaches using the retrieval results of pure diagonal matching as a reference. To this end, for increasing $K \in \mathbb{N}$, we compared each of the sets consisting of the top K matches obtained from the two index-based matching procedures with the corresponding set obtained from diagonal matching and determined the percentage of coinciding matches. Fig. 6 shows the two resulting curves (coincidence in percentage over K) for the two index-based approaches based on $\mathcal{C}_{200}$ and $\mathcal{C}_{793}$. Both approaches yield comparable numbers of coincidences—for the top seven matches the matching with the manually constructed codebook $\mathcal{C}_{793}$ correlates slightly higher with pure diagonal matching. Furthermore, a manual inspection showed that the decrease in coincidences is mainly due to noncoinciding false positives. The overall high percentage of coincidences indicate that both of the index-based approaches yield nearly the same top matches as pure diagonal matching. On the one hand, our results indicate that the learning-based approach is more *compact* in the sense of achieving a comparable retrieval quality while using a smaller codebook. On the other hand, the knowledge-based approach has the advantages of offering a larger codebook (resulting in a larger number of inverted lists and, generally, more efficient retrieval) and of being independent of training data.

D. Running Times

In our final experiments, we investigate the speedup in computation time obtained by the index-based matching in comparison to the diagonal matching procedure. All of the proposed algorithms were implemented in the MATLAB (R13) environment and the tests were run on an Athlon XP 2800+ PC with 1 GB of main memory under MS Windows 2000. To allow for a fair comparison of both methods, diagonal matching has been implemented efficiently using fast, asymptotically optimal convolution algorithms. Additionally, we have tuned the diagonal matching algorithm with respect to several particularities of the MATLAB environment.

TABLE III
AVERAGE RUNNING TIMES AND SPEEDUP FACTORS OF DIAGONAL AND INDEX-BASED AUDIO MATCHING ($\mathcal{C}_{200}$, $\lambda = 5, \theta = 0.4$) FOR THE QUERY SETS Q-R10, Q-R15, AND Q-R20

	Average running time		
	Q-R10	Q-R15	Q-R20
Diagonal matching (seconds)	11.30	11.92	12.54
Index-based matching (seconds)	0.57	0.67	0.8
Speed-up factor	**19.66**	**17.79**	**15.67**

Regarding the asymptotical running times, diagonal matching requires $O(M \log N)$ operations, where M and N denote the number of features extracted from the database and query, respectively. Assuming a bounded query length N, this results in linear asymptotic processing time of $O(M)$. The complexity of index-based audio matching using inverted lists heavily depends on the underlying data and the distribution of the lengths of the inverted files [11]. However, from other domains such as natural language retrieval, it is known that a typical inverted list-based search requires in the order of $O(M^\alpha)$ operations, for $0.4 \leq \alpha \leq 0.8$ [37]. Although a detailed analysis of the corresponding running times for the audio matching scenario is beyond the scope of this paper, our experiments clearly indicate a sublinear running time for this scenario also.

Before the actual experiments, the audio clips contained in the query sets Q-R10, Q-R15, and Q-R20 were transformed into sequences of chroma features at a feature resolution of 10 Hz (see Section II-A) and stored. The running time for computing the chroma features linearly depend on the query length and will not be of interest for the following considerations.

Table III shows the average running times (in seconds) required for audio matching performed on each of the three query sets. The running times correspond to the actual matching procedures, excluding the times for loading the database feature sequence (for diagonal matching) and for loading the inverted file index (for index-based matching), which have to be performed only once and are independent of the actual query. The running times for converting the chroma features into the eight different CENS sequences (for tempo change simulation, see Section II-B) are included as this computation is part of the query process.

The running time of the pure diagonal matching procedure (first row of Table III) is dominated by the computation of the distance function $\Delta_{\min}$—the time to determine a desired number of matches ranked according to the $\Delta_{\min}$-value is then negligible. In our experiments, we always generated a ranked list of the top 20 matches in both the index-based and the pure diagonal matching approach. In the index-based approach, we used the codebook $\mathcal{C}_{200}$ with $\lambda = 5$ and $\theta = 0.4$. Due to practical considerations, index-based query processing is implemented as follows. For each of the eight different query sequences $\mathrm{CENS}_{\mathbf{d}_j}^{\mathbf{w}_j}$, $1 \leq j \leq 8$, we determine lists of the top 40 matches satisfying $\theta \geq 0.4$ (note that a list hence may contain less than 40 matches). Subsequently, diagonal matching-based postprocessing determines the best 20 matches from the resulting candidate set of $8 \cdot 40 = 320$ matches. It turns out that the running time (second row of Table III) is dominated by the intersection and union operations on the inverted lists, whereas the running time of the postprocessing step (ranking and diagonal matching on top matches) is negligible.

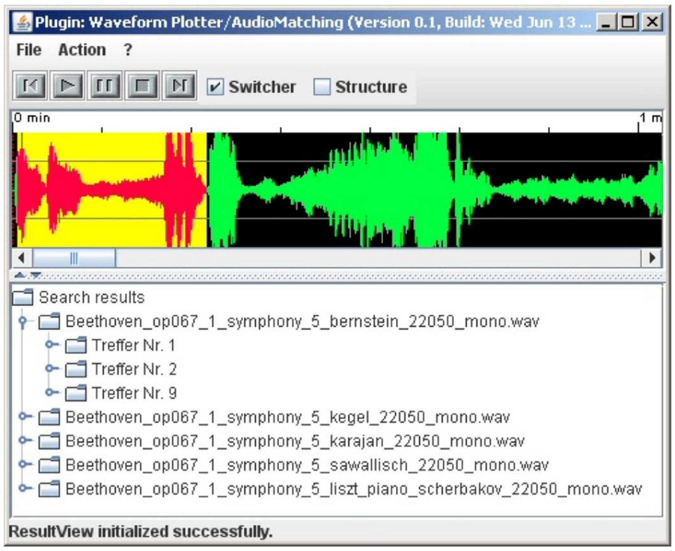
The last row of Table III shows the substantial speedup of the index-based approach as compared to pure diagonal matching which ranges from factors of about 15 to 20 depending on the query length. We finally note that larger codebooks in principle lead to smaller inverted lists, typically resulting in faster query times. On the other hand, larger threshold- and fuzzy-parameters θ and λ increase the processing time. As both effects compensate each other in our parameter settings, the processing times for using the codebooks $\mathcal{C}_{200}$ and $\mathcal{C}_{793}$ in index-based matching are roughly the same.

To conclude this section, we make some remarks regarding the scalability of the proposed index-based approach to larger databases. The application of inverted lists in the scenario of music identification shows that the success of this method depends on having a large number of inverted files available [11]. In the audio matching scenario as proposed in this paper, we have in the order of 100–1000 inverted files, which, from our experience, is suitable for efficiently handling in the order of 10 000 audio recordings. To handle larger data sets, one could either increase the number of inverted files or resort to alternative retrieval techniques such as multistage hashing.

VI. Conclusion

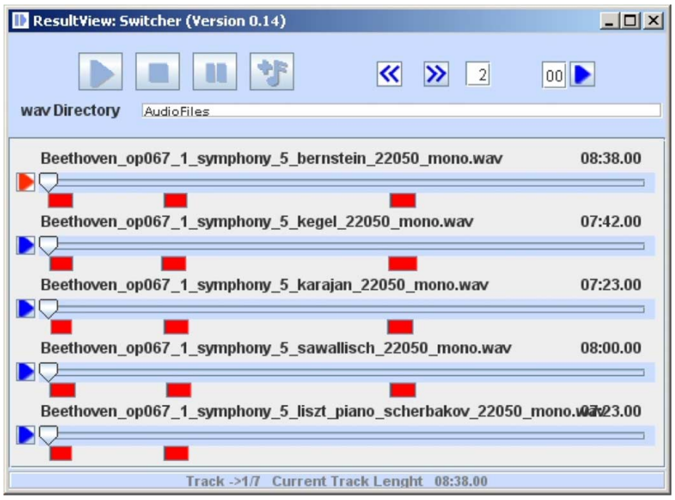
In this paper, we proposed a novel index-based audio matching algorithm, which allows for identifying and retrieving musically similar audio clips irrespective of a specific interpretation or arrangement—a task which previous audio identification algorithms cannot cope with. One simple but important idea was to already absorb a high degree of the undesired variations at the feature level by using temporally blurred chroma-based audio features. To cope with global variations in tempo and pitch, we employed the strategy of using multiple queries. Finally, we introduced a further degree of robustness into the matching process by employing fuzzy and mismatch search. The combination of these various deformation- and fault-tolerance mechanisms allowed us to employ standard indexing techniques to obtain an efficient as well as robust overall matching procedure, which can cope with significant, musically motivated variations that concern tempo, articulation, dynamics, and instrumentation. The matching procedure, in the current MATLAB implementation running on a standard PC, requires less than a second to process a 20-s query with respect to a 112-h audio database. Ongoing work is concerned with further extending the index-based matching approach to handle even larger collections of several million pieces of music, e.g., by incorporating hashing techniques [5], [8], [36]. In a hierarchical context, our index-based audio matching algorithm provides a practical and efficient component for medium-sized data sets while providing high-quality audio matches.

As a further important contribution of this paper, we discussed two strategies—a learning-based approach using a modified LBG algorithm and a knowledge-based approach—to reduce the set of the chroma-based CENS features to a small number of codebook vectors. The reduction of chroma-based features to a small codebook without loosing semantic expressiveness seems also very attractive in view of speeding up other MIR applications such as audio structure analysis [25] or audio alignment [15]. Further future work addresses the problem of


Plugin: Waveform Plotter/AudioMatching (Version 0.1, Build: Wed Jun 13 ...
File
Action
?
Switcher
Structure
0 min
1 m

Search results
Beethoven_op067_1_symphony_5_bernstein_22050_mono.wav
Treffer Nr. 1
Treffer Nr. 2
Treffer Nr. 9
Beethoven_op067_1_symphony_5_kegel_22050_mono.wav
Beethoven_op067_1_symphony_5_karajan_22050_mono.wav
Beethoven_op067_1_symphony_5_sawallisch_22050_mono.wav
Beethoven_op067_1_symphony_5_liszt_piano_scherbakov_22050_mono.wav
ResultView initialized successfully.

Fig. 7. User interface in a prototypic implementation of a query engine for audio matching. Top: the query excerpt is selected from the audio waveform (indicated by the light background). Here, the user has selected a query fragment that corresponds to our running Beethoven example, see Fig. 1(a). Bottom: all audio recordings that contain the top matches are displayed in the result list according to the ranking values. Multiple matches within the same recording are grouped together.



Beethoven_op067_1_symphony_5_bernstein_22050_mono.wav 08:38.00
Beethoven_op067_1_symphony_5_kegel_22050_mono.wav 07:42.00
Beethoven_op067_1_symphony_5_karajan_22050_mono.wav 07:23.00
Beethoven_op067_1_symphony_5_sawallisch_22050_mono.wav 08:00.00
Beethoven_op067_1_symphony_5_liszt_piano_scherbakov_22050_mono.wav 07:23.00
Track ->1/7 Current Track Lenght 08:38.00

Fig. 8. User interface for inter- and intra-document browsing of the query results of our audio matching procedure. Each audio recording that contains at least one match is displayed along with a playback button and a vertical slider. The audio subsegments that correspond to the matches are represented by boxes. In this example, the 14 boxes correspond to the top matches, see also Fig. 3.

audio matching for a broader class of music, which does not rely on characteristic harmonic progressions, by considering other features classes related to rhythm and timbre.

Finally, in an ongoing research project we are currently incorporating the audio matching functionality into the SyncPlayer system, which is an advanced audio player for multimodal presentation, browsing, and retrieval of music data [38]. In a potential application scenario, the user selects an arbitrary audio fragment as a query, see Fig. 7. Then, the audio-matching component of the system retrieves the top audio matches, which are then presented to the user via some graphical interface, see Fig. 8. Hence, audio matching not only affords novel retrieval applications, but also facilitates convenient and flexible inter- and intra-document browsing in large audio collections. Such functionalities constitute major components of future digital music libraries enabling new ways of music analysis, experience, and discovery.

ACKNOWLEDGMENT

The authors would like to thank M. Clausen for his support, C. Fremerey for his work on the SyncPlayer system, and the anonymous referees for constructive and valuable comments.

REFERENCES

[1] E. Allamanche, J. Herre, B. Fröba, and M. Cremer, "AudioID: Towards content-based identification of audio material," in *Proc. 110th AES Convention*, Amsterdam, The Netherlands, 2001, preprint 5380.
[2] P. Cano, E. Battle, T. Kalker, and J. Haitsma, "A review of algorithms for audio fingerprinting," in *Proc. 5th IEEE Workshop MMSP*, St. Thomas, Virgin Islands, 2002, pp. 169–173.
[3] M. A. Bartsch and G. H. Wakefield, "Audio thumbnailing of popular music using chroma-based representations," *IEEE Trans. Multimedia*, vol. 7, no. 1, pp. 96–104, Feb. 2005.
[4] M. Müller, F. Kurth, and M. Clausen, "Audio matching via chroma-based statistical features," in *Proc. ISMIR*, London, U.K., 2005, pp. 288–295.
[5] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," in *Proc. ISMIR*, Paris, France, 2002, pp. 107–115.
[6] F. Kurth, M. Clausen, and A. Ribbrock, "Identification of highly distorted audio material for querying large scale data bases," in *Proc. 112th AES Conv.*, Munich, Germany, 2002, preprint 5512.
[7] P. Shrestha and T. Kalker, "Audio fingerprinting in peer-to-peer networks," in *Proc. ISMIR*, Barcelona, Spain, 2004.
[8] A. Wang, "An industrial strength audio search algorithm," in *Proc. ISMIR*, Baltimore, MD, 2003, pp. 7–13.
[9] S. Baluja and M. Covell, M. M. Veloso, Ed., "Learning forgiving hash functions: Algorithms and large scale tests," in *Proc. IJCAI*, 2007, pp. 2663–2669.
[10] F. Kurth and M. Clausen, "Full-text indexing of very large audio data bases," in *Proc. 110th AES Conv.*, Amsterdam, The Netherlands, 2001, preprint 5347.
[11] M. Clausen and F. Kurth, "A unified approach to content-based and fault tolerant music identification," *IEEE Trans. Multimedia*, vol. 6, no. 5, pp. 717–731, Oct. 2004.
[12] R. Dannenberg and N. Hu, "Polyphonic audio matching for score following and intelligent audio editors," in *Proc. Int. Comput. Music Conf.*, San Francisco, CA, 2003, pp. 27–34.
[13] S. Dixon and G. Widmer, "Match: A music alignment tool chest," in *Proc. ISMIR*, London, U.K., 2005, pp. 492–497.
[14] N. Hu, R. Dannenberg, and G. Tzanetakis, "Polyphonic audio matching and alignment for music retrieval," in *Proc. IEEE WASPAA*, New Paltz, NY, Oct. 2003, pp. 185–188.
[15] M. Müller, H. Mattes, and F. Kurth, "An efficient multiscale approach to audio synchronization," in *Proc. ISMIR*, Victoria, BC, Canada, 2006, pp. 192–197.
[16] R. J. Turetsky and D. P. Ellis, "Force-aligning MIDI syntheses for polyphonic music transcription generation," in *Proc. ISMIR*, Baltimore, MD, 2003.
[17] M. Casey and M. Slaney, "Song intersection by approximate nearest neighbor search," in *Proc. ISMIR*, Victoria, BC, Canada, 2006, pp. 144–149.
[18] M. Casey and M. Slaney, "Fast recognition of remixed music audio," in *Proc. IEEE ICASSP*, 2007, pp. IV-1425–IV-1428.
[19] J. S. Downie, K. West, E. Palmpalk, and P. Lamere, 2006 [Online]. Available: http://www.music-ir.org/mirex2006/index.php/Audio_Cover_Song
[20] D. Ellis and G. Poliner, "Identifying cover songs with chroma features and dynamic programming beat tracking," in *Proc. IEEE ICASSP*, 2007, pp. IV-1429–IV-1432.
[21] E. Gómez and P. Herrera, "The song remains the same: Identifying versions of the same piece using tonal descriptors," in *Proc. ISMIR*, Victoria, BC, Canada, 2006, pp. 180–185.
[22] E. Pampalk, A. Flexer, and G. Widmer, "Improvements of audio-based music similarity and genre classification," in *Proc. ISMIR*, London, U.K., 2005, pp. 628–633.
[23] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech Audio Process.*, vol. 10, no. 5, pp. 293–302, Jul. 2002.
[24] R. Dannenberg and N. Hu, "Pattern discovery techniques for music audio," in *Proc. ISMIR*, Paris, France, 2002, pp. 63–70.
[25] M. Goto, "A chorus-section detecting method for musical audio signals," in *Proc. IEEE ICASSP*, 2003, pp. 437–440.
[26] W. Chai, "Semantic segmentation and summarization of music: methods based on tonality and recurrent structure," *IEEE Signal Process. Mag.*, vol. 23, no. 2, pp. 124–132, Mar. 2006.
[27] M. Casey and M. Slaney, "The importance of sequences in musical similarity," in *Proc. IEEE ICASSP*, Toulouse, France, 2006, pp. V-5–V-8.
[28] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes*. San Mateo, CA: Morgan Kaufmann, 1999.
[29] M. Müller and F. Kurth, "Enhancing similarity matrices for music audio analysis," in *Proc. IEEE ICASSP*, Toulouse, France, 2006, pp. V-9–V-12.
[30] M. Müller and F. Kurth, "Towards structural analysis of audio recordings in the presence of musical variations," *EURASIP J. Appl. Signal Process.*, vol. 2007, pp. 18–18, Jan. 2007.
[31] E. Zwicker and H. Fastl, *Psychoacoustics, Facts and Models*. New York: Springer-Verlag, 1990.
[32] J. R. Deller, J. G. Proakis, and J. H. L. Hansen, *Discrete-Time Processing of Speech Signals*. Washington, DC: IEEE Computer Society Press, 1999.
[33] S. R. Buss and J. P. Fillmore, "Spherical averages and applications to spherical splines and interpolation," *ACM Trans. Graph.*, vol. 20, no. 2, pp. 95–126, 2001.
[34] G. Navarro, "A guided tour to approximate string matching," *ACM Comput. Surv.*, vol. 33, no. 1, pp. 31–88, 2001.
[35] G. Navarro, R. A. Baeza-Yates, E. Sutinen, and J. Tarhio, "Indexing methods for approximate string matching," *IEEE Data Eng. Bull.*, vol. 24, no. 4, pp. 19–27, 2001.
[36] H. Wolfson and I. Rigoutsos, "Geometric hashing: An overview," *IEEE Comput. Sci. Eng.*, vol. 4, no. 4, pp. 10–21, 1997.
[37] M. Araujo, G. Navarro, and N. Ziviani, R. Baeza-Yates, Ed., "Large text searching allowing errors," in *Proc. 4th South Amer. Workshop String Process.*, Valparaiso, Chile, 1997, pp. 2–20 [Online]. Available: citeseer.ist.psu.edu/araujo97large.html
[38] F. Kurth, M. Müller, D. Damm, C. Fremerey, A. Ribbrock, and M. Clausen, "SyncPlayer—An advanced system for content-based audio access," in *Proc. ISMIR*, London, U.K., 2005.

Frank Kurth (M'04) received the M.S. and Doctor of Natural Sciences (Dr.rer.nat.) degrees in computer science from Bonn University, Bonn, Germany, in 1997 and 1999, respectively.

Since his Habilitation (postdoctoral lecture qualification) in computer science in 2004, he holds the title of a Privatdozent and is lecturing at Bonn University. Currently, he is with the Communication Systems Group Research Institute for Communication, Information Processing and Ergonomics (FKIE), Research Establishment for Applied Scenice (FGAN), Wachtberg-Werthhoven, Germany. His research interests include audio signal processing, fast algorithms, multimedia information retrieval, and digital libraries for multimedia documents. Particular fields of interest are music information retrieval, fast content-based retrieval, and bioacoustical pattern matching.

Meinard Müller received the M.S. and Doctor of Natural Sciences (Dr.rer.nat.) degrees in computer science from Bonn University, Bonn, Germany, in 1997 and 2001, respectively.

In 2002 and 2003, he conducted postdoctoral research in combinatorics at the Mathematical Department, Keio University, Tokyo, Japan. Currently, he is a member of the Max-Planck Institute for Computer Science, Saarbrücken, Germany, working as a Senior Researcher at the Max-Planck Center for Visual Computing and Communication. His research interests include digital signal processing, multimedia information retrieval, computational group theory, and combinatorics. His special research topics comprise audio signal processing, computational musicology, analysis of 3-D motion capture data, and content-based retrieval in multimedia documents.