Searching musical audio using symbolic queries

Suyoto, Iman S. H.; Uitdenbogerd, Alexandra; Scholer, Falk

Suyoto, I. S. H., Uitdenbogerd, A., & Scholer, F. (2008). Searching musical audio using symbolic queries. IEEE Transactions on Audio, Speech and Language Processing, 16(2), 372–381. https://doi.org/10.1109/TASL.2007.911644

Published Version: https://doi.org/10.1109/TASL.2007.911644

Searching Musical Audio Using Symbolic Queries

Iman S. H. Suyoto, Alexandra L. Uitdenbogerd, and Falk Scholer

***Abstract*—Finding a piece of music based on its content is a key problem in music information retrieval. For example, a user may be interested in finding music based on knowledge of only a small fragment of the overall tune. In this paper, we consider the searching of musical audio using symbolic queries. We first propose a relative pitch approach for representing queries and pieces. Experiments show that this technique, while effective, works best when the whole tune is used as a query. We then present an algorithm for matching based on a pitch classes approach, using the longest common subsequence between a query and target. Experimental evaluation shows that our technique is highly effective, with a mean average precision of 0.77 on a collection of 1808 recordings. Significantly, our technique is robust for truncated queries, being able to maintain effectiveness and to retrieve correct answers whether the query fragment is taken from the beginning, middle, or end of a piece. This represents a significant reduction in the burden placed on users when formulating queries.**

***Index Terms*—Dynamic programming, information search and retrieval, query formulation, retrieval models, search process, sound and music computing.**

I. Introduction

THE WAY that users find music has changed considerably in the last decade, with much music available online. While music is often located by artist name, song title, or lyrics, there are many times when users wish to find a particular tune by content.

In the field of music information retrieval, the desired objective is the ability to retrieve music from an audio collection given a query, representing a portion of the music, that is either sung, played, or otherwise encoded. To date, most practical systems that retrieve music given a melody fragment as a query, are restricted to music collections that are in a symbolic format such as MIDI. The major difficulty lies in the automatic transcription of recorded music, due to the complexity of the audio signal when there is more than one note sounding simultaneously, multiple musical instrument timbres, and a complex acoustic environment.

Monophonic audio can be automatically transformed into a sequence of symbols more accurately than polyphonic audio. In the context of singing, about 80% of notes can be accurately identified [3]. Because of the perplexing factors of timbre (the sound of the instruments [7, p. 334]) and the acoustic environment where the music is performed, automatic polyphonic audio transcription is harder. Those factors cause harmonics that can be erroneously perceived as actual notes. For instance, the TS-AudioToMidi software package that we use in our work, in its default configuration, on average mistakenly adds about three times as many notes to the symbolic representation of a piece of music than actually exist. Moreover, some actual notes are missing in the resulting transcriptions.

In this paper, we test the feasibility of finding polyphonic audio recordings of music given a symbolic representation of the same work (query by score). There are two tasks presented here: audio retrieval using its *full* symbolic equivalent and audio retrieval using its *truncated* symbolic equivalent. We investigate a technique that makes use of dynamic programming on sequences of relative pitches. An example scenario where this can be useful is when a user has a MIDI file and wishes to obtain a recorded performance of the song. The technique is explained in Section III. We also propose a new technique based on score-to-audio alignment that can be used to retrieve an audio collection using short queries, instead of full scores. This makes use of dynamic programming on sequences of pitch classes, as is explained in Section IV. In Section V, we discuss our experimental setup, and the results and discussions are presented in Section VI. Finally, we present our conclusions and suggestions for future work in Section VII.

The experimental results show that our techniques are highly effective for a collection of about 1800 audio pieces, the rank of most queries is high when given a symbolic query, even when the query is far shorter than its full version. We have shown that it is possible to use current transcription technology to retrieve answers to symbolic queries effectively.

Accompanying materials for this paper can be found at http://mirt.cs.rmit.edu.au/pubs/sd.

II. Related Work

Out of the two tasks in this work, the full-query task is related to score-to-audio alignment. While both tasks may share similar techniques, the objectives of the tasks differ, in spite of the same interest in matching an audio file with its symbolic equivalent. A retrieval task such as ours aims to rank correct answers highly. As such, the ability to separate correct answers from incorrect ones is crucial. In score-to-audio alignment, success is typically determined by how many notes are correctly or incorrectly identified (as shown elsewhere [19], [26], [29], [32]), an aspect that is not necessarily (or at least, solely) considered for retrieval effectiveness. As an implication, structural match is also a matter of concern in score-to-audio alignment, whereas approximate structural matches are tolerated in a retrieval task.

Previously, Pickens *et al.* [22] proposed a harmonic modeling approach for matching polyphonic audio queries against a polyphonic symbolic collection (this is similar to our full-query task). The queries consisted of human performance recordings and audio synthesized from MIDI. Their approach achieved

Manuscript received December 19, 2006; revised September 24, 2007. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Malcolm Slaney.

The authors are with the School of Computer Science and Information Technology, Royal Melbourne Institute of Technology, Melbourne, Victoria 3001, Australia. (e-mail: iman.suyoto@rmit.edu.au; alexandra.uitdenbogerd@rmit.edu.au; falk.scholer@rmit.edu.au).

Digital Object Identifier 10.1109/TASL.2007.911644

mean average precision (MAP) of 0.479. We take a different approach in this work. Here, we match symbolic queries against an audio collection. Transcription is applied against the files in our collection, whereas they use transcription for their queries. Statistical patterns of chords in music were considered in their matching process, while they are not in ours.

Hu *et al.* [12] also investigated matching polyphonic audio queries against a polyphonic symbolic collection using chroma.[1] Their collection contained 259 MIDI files and they used 51 audio queries. Both the collection and queries were songs by the Beatles. The pieces in the collection were synthesized into audio, from which the chroma were extracted. Their method achieved mean precision at 1 of 0.49. In our work, pitch classes are extracted directly from both the queries and the pieces in the collection (after transcription, which introduces extraneous notes, see Section III-A). We also use a much bigger collection.

Aligning a polyphonic audio collection with monophonic symbolic queries has been considered by Shalev-Shwartz *et al.* [28] using a probabilistic approach. They used temporal and spectral features with a collection of 832 one-minute opera performances with orchestra accompaniment and 50 queries. This differs from our approach in that we do not truncate any of the 1808 pieces in our collection.

It has been shown that specific recordings can be retrieved using a query of the same version but with different audio quality [39]. Audio signatures are used for this purpose. Cover or alternate versions of music are not normally found, however. This technique is implemented in the Shazam system.[2] To match cover versions, other techniques have been proposed. Some early work explored the use of long-term structure, such as (musical) dynamics [6] or timbral texture [1], [2]. However, the scalability of these techniques is still subject to further research, as the collections used in the experiments contained less than 100 pieces. Matching MIDI files with monophonic hummed queries have proven to be more successful [4], [11], [13], [17], [30]. Song *et al.* [31] explored retrieval from a polyphonic audio collection with hummed queries by extracting sets of possible notes from the audio. The collection contained 92 melody clips of about 15–20 s long, and there were 176 queries. From ten trials, correct music was retrieved within the top ten about seven times. The Midomi system[3] is designed to only match human voice to human voice.

Audio-to-audio matching has recently been used for cover version retrieval. With a collection of just 90 pieces, the cover version detection approach using chroma suggested by Gómez and Herrera [8] yielded accuracy of around 50%. Marolt [16] attempted using a combination of melody-based and chroma representations on a collection of 1820 pieces with 36 of them used as the queries. An effectiveness of 27% of hits in the top five returned answers was achieved.

A variation of the longest common subsequence (LCS) algorithm has been attempted, however, for monophonic music matching [9]. Variations in speed and inaccuracies in rhythm were simulated, and the LCS algorithm was modified to handle such problems. The best technique obtained almost 90% of correct answers ranked in the top five. However, our results show that the LCS algorithm does not need to be modified for the effectiveness of our tasks.

Matching polyphonic symbolic queries with a polyphonic symbolic collection can be considered mostly solved [21], [23], [35], [37]. However, our current work shows that a technique that works effectively for matching polyphonic symbolic queries with polyphonic symbolic collections does not perform as effectively for polyphonic audio collections. We show this problem in this paper and also devise an alternative approach that works effectively.

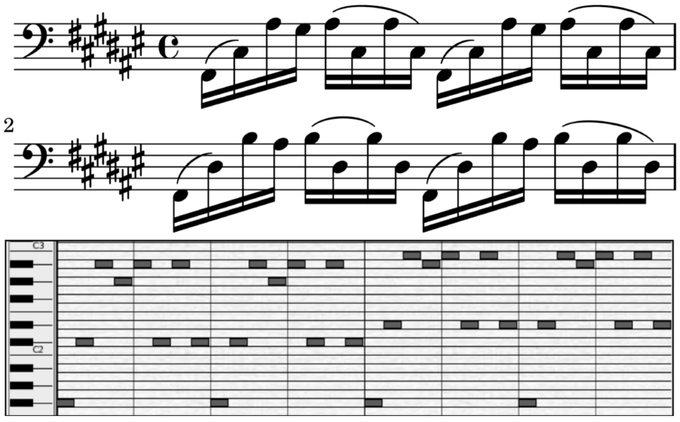

Fig. 1. First two bars of BWV 1007 Prelude. This piece is composed in G major. However, in this figure, it is transposed down by one semitone to match the transcription (see Fig. 2) of a performance using the Baroque tuning (A = 415 Hz).

III. Relative Pitch Approach

In Section II, we surveyed successful research in polyphonic symbolic retrieval using polyphonic symbolic queries. Relative pitch representation has been shown to work very well for this task [37]. This section discusses our approach, which is adjusted for the task of retrieving polyphonic audio using its symbolic equivalent.

The matching process is composed of five stages: transcription (of audio to symbolic data), noise removal, bass-part extraction, standardization, and alignment. These are discussed below.

A. Transcription

In the transcription stage, the audio files in the collection are transcribed so that symbolic data are obtained. The transcriber we use is TS-AudioToMidi 3.30.[4] The transcription results are saved in the Standard MIDI file format.[5]

Current transcription technology still produces musical symbols with much noise in the form of extraneous notes, particularly when the audio consists of more than one timbre and is polyphonic. As an example, one of the tracks in our collection is J. S. Bach's BWV 1007 Prelude performed by Carrai. The first two bars of the track are shown in Fig. 1. The transcription result is shown in Fig. 2. It contains many extraneous notes. On

[1]The terms "chroma" and "pitch classes" can often be used interchangeably, e.g., in Müller *et al.* [18]. Audio signal analysts usually use the former while music theorists usually use the latter.

[2]See http://www.shazam.com.

[3]See http://www.midomi.com.

[4]See http://audioto.com/eng/aud2midi.htm.

[5]See http://www.midi.org/about-midi/abtmidi2.shtml.

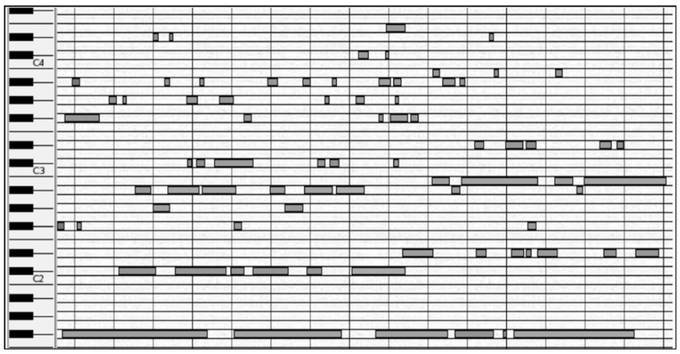

Fig. 2. First few notes of the transcription result of BWV 1007 Prelude performance by Carrai that corresponds to the first two bars of the score shown in Fig. 1.

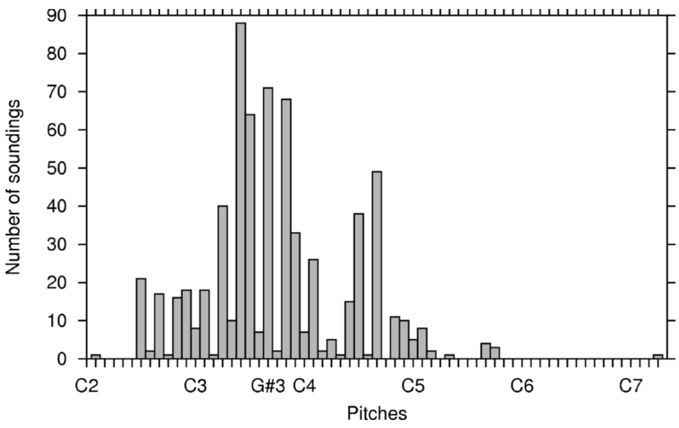

Fig. 3. Pitch distribution of the transcription result of J. S. Bach's BWV 1007 Prelude performed by Carrai. G#3 is the median. Only notes with MIDI velocities of equal to or greater than V_t are included.

average, for one actual note, about three notes are produced in the transcription.

B. Noise Removal

As mentioned previously, current transcription technology leaves much noise in its symbolic output. Therefore, we need a noise removal procedure to help us generate a retrievable sequence.

We use a noise removal heuristic that depends on the statistics of a tune, particularly that of pitch. The result of this process is not intended for listening purposes but for producing "clean" symbolic sequences that can be used for retrieval.

The first step in filtering a tune is removing notes that are perceived as "too soft." A softness threshold V_t, where V_t is a valid MIDI velocity value, is used here. Any notes softer than V_t are discarded. The next step is building a histogram of pitches. As an example, the histogram showing the number of soundings of pitches in the transcription result of J. S. Bach's BWV 1007 Prelude performed by Carrai (excerpt shown in Fig. 2) is plotted in Fig. 3. By building a histogram, the pitch median can be determined. Let us call the pitch median $\tilde{P}$. For each pitch p, if $p > \tilde{P}$, it is removed. The filtered result is shown in Fig. 4.

This heuristic was developed through observation of typical transcription results. Harmonics due to instrument timbre and

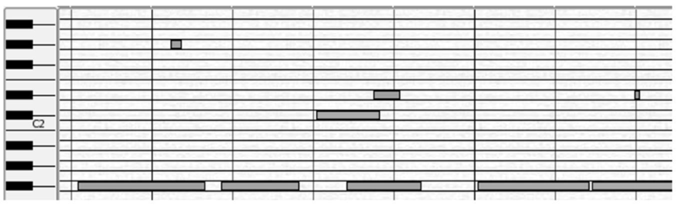

Fig. 4. Filtered result of BWV 1007 Prelude performance by Carrai. Note that although as shown in Fig. 2 the first note is an F#3, it does not appear here because it is softer than V_t.

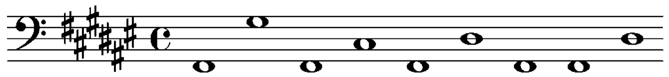

Fig. 5. Bass-line extraction result of the tune shown in Fig. 4.

reverberation effects tend to lead to additional notes being transcribed along with the actual melody pitch. The problem of finding melody lines is still only partially solved.[6]

C. Bass-Part Extraction

From every transcribed tune, a monophonic sequence that represents the tune is extracted. Uitdenbogerd and Zobel [37] show that the ALL-MONO algorithm is the most effective melody extraction technique for retrieval purposes. When there is a note or a chord starting to sound, the note having the highest pitch becomes "the melody note." If there is a note m of length l_m sounding at time t_m and another note n sounding at t_n so that $l_m + t_m \geq t_n$, then l_m will become $l'_m \leftarrow l_m - (t_n - t_m)$. In other words, note overlaps are removed. For our experiments, we modify the ALL-MONO algorithm so that instead of taking the highest pitch (which works well for melody extraction), we take the lowest pitch as we intend to extract the bass part. In Algorithm 1 we show the bass-part extraction algorithm we use (note duration information however is not extracted, as this is not used in post-extraction stages). This process is applied to tunes in the collection and also the query tune. The bass part extracted from Fig. 4 is the sequence of notes shown in Fig. 5. Note that duration information is not extracted in this step. Consequently, rhythmic patterns have also been removed.

Algorithm 1 Bass-part extraction algorithm. A note is expressed as a tuple $n = \langle p, o \rangle$ where p is the pitch and o is the onset time. The base index is 0. P is the sequence of the representative bass part. "π_x" is the relational operator for projecting the x attribute.

```
Require: array of notes $\mathbf{N}$
    Sort $\mathbf{N}$ by ascending onset time as the first sort key and
    descending pitch as the second sort key.
    [Start taking the lowest note at any onset time.]
    for $i = 0 \ldots |\mathbf{N}| - 2$ do
        if $(\pi_o n_i \neq \pi_o n_{i+1})$ then
            Append $\pi_p n_i$ to $P$.
        end if
    end for
    Append $\pi_p n_{|\mathbf{N}|-1}$ to $P$.
    [End.]
    return $P$.
```

[6]Research on this area is active, as described elsewhere [5], [14], [15], [20], [24], [25].

The rationale of modifying the ALL-MONO algorithm is that high pitches are not sufficiently reliable since most extraneous notes are high in pitch. Also, in classical music (particularly from older periods such as Baroque), the lower pitches are quite melodic, for example, J. S. Bach's Inventions (Baroque period), Mozart's piano sonatas (Classical period), and Chopin's fantasies (Romantic period), compared to three-chord blues tunes, where the chord sequence (and thus the bass line) follows a common pattern.

D. Standardization

Once a monophonic "melody" has been acquired, we can generate a string that represents it. The idea is to match the string that represents the query tune with the sequences that represent the tunes in the collection. To facilitate ranking, there must be a measure that reflects how similar two matched strings are to each other. This is achieved using the score returned by an approximate matching function.

A standardization that has been shown to work well is the *directed modulo-12* standardization. In this standardization, a note is represented as a value R which is the interval between the current note and its previous note, scaled to a maximum of 12 semitones (one octave) [33], [36]

$$R \equiv d(1 + ((\Delta - 1) \bmod 12)) \tag{1}$$

where Δ is the interval between a note and its previous note (absolute value) and d is $+1$ if the previous note is lower than the current note, -1 if higher, and 0 if otherwise. Using the directed modulo-12 standardization, the melody shown in Fig. 5 is represented as $\langle +2, -2, +7, -7, +9, -9, 0, +9 \rangle$.

The clear advantage of using this standardization is transposition-invariance. For example, the melodies C4-E4-G4 and G3-B3-D4 match perfectly, as both are represented as $\langle +4, +3 \rangle$.

E. Alignment

For the alignment phase of our approach, we apply dynamic programming to the pitch sequences, a technique that has been previously applied to symbolic music matching [4], [33], [37] as well as audio [1], [6]. To investigate the ability of manually constructed symbolic sequences being matched with transcription-of-audio sequences, we use *global alignment* with custom operation weights, which calculates the number of edit operations (namely *match*, *mismatch*, and *insertion/deletion*) to transform one string to another [10, p. 216–217, 224]. The query and the target tune(s) are complete albeit different renditions of the same tune. However, we also inspect the possibility of using *local alignment*, which discovers similar regions between two strings [10, p. 230–223] to find out whether it is possible to issue a query in the form of a short piece of the target tune. We use both algorithms in our experiment (see Section V) with sequences of directed modulo-12 representation [see (1)] of the tunes as the strings.

IV. Pitch Classes Approach

We conducted preliminary experiments with the relative pitch approach suggested in Section III. While some queries successfully retrieved correct answers in the first rank position, we felt that the results still needed much improvement (the results are presented in Section VI).

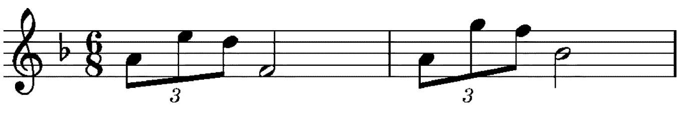
Fig. 6. Melody from "When the Sun Sets II" by ade ishs. Using pitch classes, it is represented as "A E D F A G F B♭."

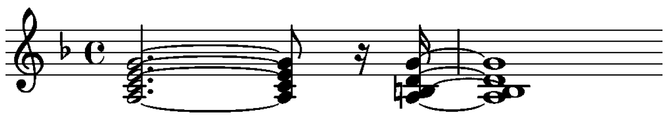
Fig. 7. Chords from "Jakarta at Night I" by ade ishs. Using pitch classes and "flattening," it is represented as "A C E G A B D G."

Our previous work [34] has shown that it is possible to effectively retrieve audio files using their symbolic equivalent when the notes in both the transcribed audio and symbolic sequences are represented as pitch classes. The similarity measurement was the LCS score, which is the count of symbols common in two strings based on the sequence of their occurrences (symbols are not necessarily adjacent) [10, p. 227–228], normalized by the square of the natural logarithm of the answer. Using the same collection and query set as we use here, a MAP value of 0.826 was achieved. In this paper, we focus on retrieval using truncated queries.

The matching process is composed of three stages: transcription, standardization, and alignment. The transcription stage here is the same as the transcription stage explained in Section III-A, so we only discuss the standardization and alignment stages in the following subsections.

A. Standardization

The previous transcription stage produces MIDI files. These are now converted into strings of pitch classes for approximate matching, i.e., a pitch is represented as its pitch name without the octave. Rest notes are discarded. For example, the melody shown in Fig. 6 is represented as "A E D F A G F B♭."

A chord is "flattened" so that it becomes an arpeggio. This is done by ordering the notes in the chord from the lowest *actual* pitch to the highest. For example, the score in Fig. 7, is represented as "A C E G A B D G."

An implication of using pitch classes to represent the notes is that the representation lacks transposition-invariance. This forces the alignment process to transpose queries to every possible key. We explain this in greater detail in Section IV-B. Although this is less efficient than the relative pitch approach discussed in Section III, we shall see in Section VI that much higher retrieval effectiveness is gained.

B. Alignment

Like the alignment stage described in Section III-E, the alignment stage here also applies dynamic programming. However, we additionally make use of a sliding window technique here.

The current task is designed to solve the problem when a user's query is only a short part of a song. We have previously experimented with the approach used to effectively retrieve audio using its full symbolic equivalent [34]. However, this was shown to be ineffective when the queries are much shorter than the target piece, as the previous approach made the assumption that the length of the target pieces is somewhat proportional to the query length. Therefore, a sliding window technique is more appropriate.

Our method makes use of a window size parameter d. If q is the query, we define a window size function W

$$W \equiv \lceil 2d|q| \rceil \tag{2}$$

and $W+1$ is the actual window size itself. Matching starts from the beginning of an answer a. We take a substring of a: let $z = a_0 \ldots a_W$. From here, a candidate score s is produced by computing the LCS score between z and q. After that, the window slides by $\lceil d \rceil$ positions. Now z becomes $a_{\lceil d \rceil} \ldots a_{\lceil d \rceil + W}$. A new candidate score s' is produced by computing the LCS score between z and q again. We then compare the two candidate scores s and s', and the higher one becomes the new value of s. Again, the window slides by $\lceil d \rceil$. This process continues while $n\lceil d \rceil + W < |a|$, where n is a nonnegative integer. The query has to be matched for all possible keys. Therefore, it is transposed up by one semitone, and the above process is repeated for all keys, resulting in 11 transpositions. Let us define $S_d(q, a, d)$ to be the similarity score between query q and answer a with the parameter d; the function is set to the final value of s. The (unoptimized) algorithm is shown in Algorithm 2.

Algorithm 2 The algorithm to calculate $S_d(q, a, d)$. $t(q, s)$ is a function transposing q by s semitones ($s \in \{0, 1, 2, \ldots, 11\}$). $\mathrm{LCS}(T, a)$ is the LCS score between T and a. We use 0 as the base index.

```
Require: query $q$, answer $a$, parameter $d$
  $W \leftarrow \lceil 2d|q| \rceil$
  $s \leftarrow 0$
  for $k = 0 \ldots 11$ do
    $L \leftarrow 0$
    $q' \leftarrow t(q, k)$
    while $L + W < |a|$ do
      $s' \leftarrow \mathrm{LCS}(q', a_L \ldots a_{L+W})$
      if $s' > s$ then
        $s \leftarrow s'$
      end if
      $L \leftarrow L + \lceil d \rceil$
    end while
  end for
  return $s$.
```

As an example of how to apply Algorithm 2, consider a query $q =$ "G A C" (hence $|q| = 3$) and it is to be aligned with the answer $a =$ "C G E D G# G# E A# C# D D# C# D" (hence $|a| = 13$). Suppose that we choose $d = 1.2$, hence $W = \lceil 2 \times 1.2 \times 3 \rceil = 8$, hence the window size of 9. The current window is boxed.

$$\boxed{\texttt{C G E D G\# G\# E A\# C\#}}\ \texttt{D D\# C\# D.}$$

The LCS score between "G A C" and the boxed substring is 1. This becomes s. The window then slides by $\lceil 1.2 \rceil = 2$:

$$\texttt{C G}\ \boxed{\texttt{E D G\# G\# E A\# C\# D D\#}}\ \texttt{C\# D.}$$

The LCS score between "G A C" and the boxed substring is 0. The candidate score s is now still 1 as $1 > 0$. The window then slides by 2 again:

$$\texttt{C G E D}\boxed{\texttt{G\# G\# E A\# C\# D D\# C\# D}}\,.$$

The LCS score between "G A C" and the boxed substring is 0. s remains unchanged. Next, the query is transposed by one semitone, becoming $q =$ "G# A# C#". The window position is rewound.

$$\boxed{\texttt{C G E D G\# G\# E A\# C\#}}\ \texttt{D D\# C\# D.}$$

The LCS score between "G# A# C#" and the boxed substring is 3. The candidate score s is now 3 as $3 > 1$. This is repeated until q is transposed up to 11 times.

V. Experimental Setup and Evaluation Methodology

As our experimental collection, we use the classical music collection of Magnatune[7] (as af April 28, 2005), stored as MP3 (MPEG Layer 3)[8] files. Multiple versions of some pieces exist in the collection. For example, three different artists perform J. S. Bach's Suite I for Cello Solo (BWV 1007). The audio files were transcribed using TS-AudioToMidi with its default settings. It failed to process some files. In total, we obtained 1808 transcriptions stored as MIDI files.

We formed our query set by gathering the MIDI versions of some of the works in the collection. Our sources are the Mutopia Project,[9] Kern Scores,[10] and MuseData.[11] In total, there are 34 queries, all having relevant answers in the collection. The same queries are used for all tasks, except we truncated the queries for the short-query task. Details of the queries are presented in Table I.

Various instrumentations, including orchestra, are used in the target pieces. Two out of the 34 queries are completely monophonic, but the compositions are originally also monophonic. However, since the noisy transcription process produces polyphonic transcriptions, even from monophonic pieces (as a real example, see Figs. 1 and 2), the difficulty of retrieval still holds.

[7]See http://www.magnatune.com/genres/classical/.

[8]See http://www.iis.fraunhofer.de/amm/techinf/layer3/.

[9]See http://www.mutopiaproject.org.

[10]See http://kern.humdrum.net.

[11]See http://www.musedata.org.

TABLE I
QUERIES AND THE NUMBER OF RELEVANT COVERS IN THE COLLECTION (C). (KS: KERN SCORES, MD-1: MUSEDATA [OPTIMIZED FOR PRINTING], MD-P: MUSEDATA [OPTIMIZED FOR LISTENING], MP: THE MUTOPIA PROJECT.)

Query	Title	Source	C
B1011_C	BWV 1011 Courante	KS	3
B1011_S	BWV 1011 Sarabande	KS	3
COR_O1N7_1	Corelli, Trio Sonata Opus 1 No. 7 in C Major (Movement 1)	KS	1
COR_O1N7_2	Corelli, Trio Sonata Opus 1 No. 7 in C Major (Movement 2)	KS	1
COR_O1N7_3	Corelli, Trio Sonata Opus 1 No. 7 in C Major (Movement 3)	KS	1
DUF_ACB	Dufay, Adieu Ces Bons Vins De Lannoys	KS	1
JOP_STOPTIME	Joplin, Stoptime Rag	KS	1
K545_1	K 545 Movement 1	KS	1
K545_2	K 545 Movement 2	KS	1
K238	K 238	KS	1
B870_F_MD1	BWV 870 Fugue	MD-1	1
B870_P_MD1	BWV 870 Prelude	MD-1	1
B870_F_MDP	BWV 870 Fugue	MD-P	1
B870_P_MDP	BWV 870 Prelude	MD-P	1
B1007_A	BWV 1007 Allemande	MP	3
B1007_C	BWV 1007 Courante	MP	3
B1007_G	BWV 1007 Gigue	MP	4
B1007_M	BWV 1007 Menuets	MP	3
B1007_P	BWV 1007 Prelude	MP	3
B1007_S	BWV 1007 Sarabande	MP	3
B1010_A	BWV 1010 Allemande	MP	3
B1010_B1	BWV 1010 Bourrée I	MP	3
B1010_B2	BWV 1010 Bourrée II	MP	3
B1010_C	BWV 1010 Courante	MP	3
B1010_G	BWV 1010 Gigue	MP	3
B1010_P	BWV 1010 Prelude	MP	3
B1010_S	BWV 1010 Sarabande	MP	3
B1042_AD	BWV 1042 Adagio	MP	1
B1042_AL	BWV 1042 Allegro	MP	1
B1042_AA	BWV 1042 Allegro Assai	MP	1
B846_F	BWV 846 Fugue	MP	1
B846_P	BWV 846 Prelude	MP	1
B860_F	BWV 860 Fugue	MP	1
B860_P	BWV 860 Prelude	MP	1

J. S. Bach composed BWV 1007 Menuets as two parts. Our audio collection contains three covers, with both parts stored in a single file. Originally, we obtained the symbolic version from the Mutopia Project with both parts separated. We concatenated these to construct a single query. Bach also composed BWV 1010 Bourrées as two parts. Again, our audio collection contains three covers, with both parts stored in a single file. However, we used two queries, each for BWV 1010 Bourrée I and BWV 1010 Bourrée II. We use these differences to examine the robustness of our full-query methods in such cases. BWV 1042 in the collection is polyphonic whereas the queries (B1042_AD, B1042_AL, and B1042_AA) are monophonic. We shall see whether our approaches work in this condition.

The queries in the set differ greatly in length. This is visualized in Fig. 8. The average number of symbols across the set is 814.15.

In our experiment with the relative pitch approach (Section III), we used $V_t = 48$ for filtering[12] (see Section III-B). This was applied to both the tunes in the collection and the query tunes. We tested four alignment procedures.

- Global alignment on whole tunes and whole queries. We call this SBDG.

[12]We have done experiments with a few values and $V_t = 48$ came out as the best. We have tried to inspect whether there is a pattern for a good V_t in a few pieces, but have so far not found a method for automatically choosing parameter values.

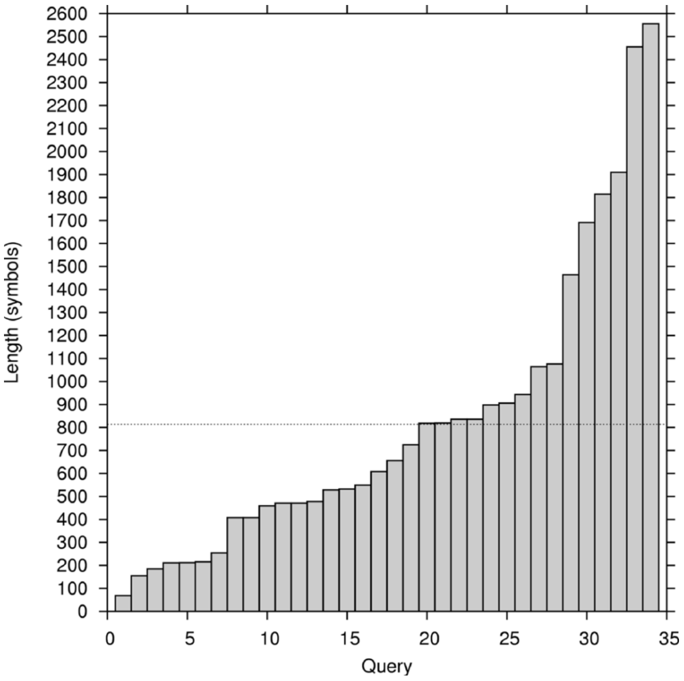

Fig. 8. Distribution of pretruncation query lengths. The dashed line indicates the average.

- Local alignment on whole tunes and whole queries. We call this SBDL.

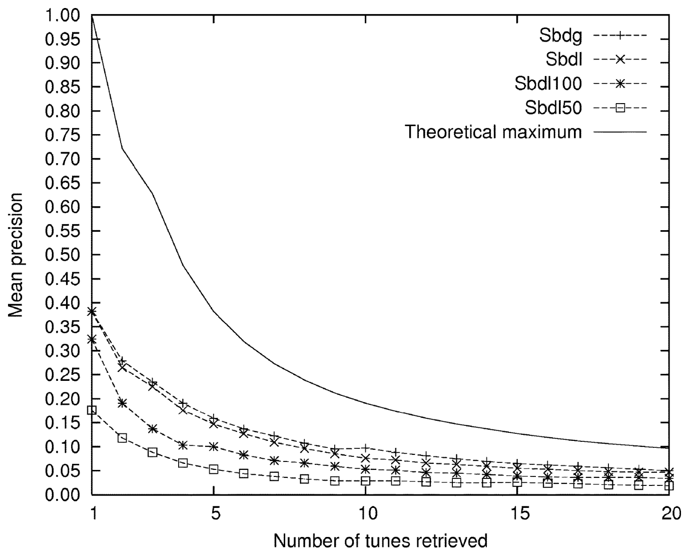

Fig. 9. Mean precision at N ($\langle P_N \rangle$) curves for SBDG, SBDL, SBDL100, and SBDL50.

- The query tunes were truncated to a maximum of 50 symbols (if q is the original query, then $q_0 \ldots q_{49}$ is used) and matched using local alignment. We call this SBDL50.
- The query tunes were truncated to a maximum of 100 symbols (if q is the original query, then $q_0 \ldots q_{99}$ is used) and matched using local alignment. We call this SBDL100.

The alignment parameters we use for match, mismatch, and insertion/deletion operations are 10, -1, and -2, respectively (see Section III-E). This combination of values was determined by preliminary experiments.

To evaluate the effectiveness of our approaches, we use measures based on the *precision* of our retrieval techniques; precision is defined as the proportion of retrieved documents that are relevant. Given the nature of the retrieval task considered in this work—finding matching pieces of music in response to a query—users are most likely to be interested in matches that occur high in the ranked list of returned answers. Our analysis therefore focuses on the precision at a cutoff value N, P_N, which measures the number of relevant answers that have been returned in the first N positions of the ranked list. To give an overall measure of performance, we also report the widely used MAP measure. The average precision for a single query is the mean precision at each relevant answer that is found; MAP is then the mean of the average precision scores over a run of queries [38].

In our analysis, we make use of statistical significance tests to investigate whether differences in results are caused by true underlying effects rather than random variation. Unless otherwise indicated, we use a two-tailed paired t-test, which has been shown to be effective for differentiating between runs in information retrieval environments [27].

VI. Results and Discussion

A. Relative Pitch Approach

The results for our experiments with the methods mentioned in Section III are shown in Fig. 9. Based on the mean precision values in the figure, the most effective method is SBDG, followed by SBDL, both achieving $P_1 = 0.382$, and then SBDL100 ($P_1 = 0.324$), and finally SBDL50 ($P_1 = 0.176$). SBDG seems to be just marginally more effective than SBDL (P_{10} is equal to 0.097 for SBDG and 0.076 for SBDL). Using SBDG, 53% of all queries successfully retrieved an answer in the top 10, whereas with SBDL, 47% of all queries successfully retrieved an answer in the top 10. Using SBDL50, 26% of all queries retrieved an answer in the top 10, and 32% in the top 20 of returned results, whereas with SBDL100, 47% of all queries retrieved an answer in the top 10, and 53% in the top 20. The P_N values show that shorter queries tend to sacrifice effectiveness.

TABLE II
MAP Values for SBDG, SBDL, SBDL100, and SBDL50. The Best Value is Highlighted

Method	MAP
SBDG	0.374
SBDL	0.340
SBDL100	0.229
SBDL50	0.121

The overall performance of the methods as measured by MAP is shown in Table II. To find out whether SBDG, the best performing method, is significantly better than SBDL, the second best method, we conduct a paired t-test. The result is that $p > 0.05$, meaning that there is no evidence that SBDG is significantly better than SBDL. However, using the same test on SBDG and the other two techniques, it is shown that SBDG is statistically significantly better ($p < 0.01$).

The retrieval performance for BWV 1007 was among the best. Using SBDG, all of the queries produced the first correct answer in the first place, except B1007_M, which produced the first correct answers at rank 2. All of the queries retrieved at least two correct answers in the top 10. They also retrieved all correct answers in the top 20 except B1007_M, which only retrieves two of them (at ranks 2 and 3, the other one at rank 35). Using SBDL, all of the queries produced the first correct answer at the first place with no exception. Only B1007_S failed to retrieve all correct answers in the top 20.

The polyphonic piano pieces BWV 846, BWV 860, and K 545 were also retrieved in the first place using SBDG. The retrieval performance SBDL is the same except for B846_F, which produces the correct answer at rank 3. These good results may be partly due to the relatively low amount of noise in the transcriptions. The ratio of the number of notes in the transcriptions to the number of notes in the query is less than 1.3 for those pieces.

The retrieval performance for BWV 1010 is only good for B1010_A and B1010_P. For B1010_A, the correct answers were retrieved at ranks 2, 7, and 93 (using SBDG) and 3, 19, and 29 (using SBDL). For B1010_P, all the three covers were retrieved in the top 3. For B1010_C and B1010_S, both failed to retrieve answers in the top 20 using any of the methods. B1010_B2 with SBDL however retrieved a correct answer in the second rank position.

Retrieval performance was poor for DUF_ACB, JOP_STOPTIME, B1042_AD, B1042_AL, and B1042_AA using all methods. All of them could not retrieve a correct answer in the top 90. B1042_AD could not even retrieve a correct answer in the top 1000. This is actually not surprising since the query contains the high pitches and lacks the "bass part."

B1011_C performed poorly, but B1011_S retrieved the correct answer in the first place using all methods.

Referring to BWV 1010 Gigue and BWV 1011 Sarabande, it should be noted that even though an audio performance is monophonic, its transcription result is polyphonic (Figs. 1 and 2 serve as an example).

All the mostly or fully polyphonic audio, with the exception of the piano pieces, were hard to retrieve. Our simple filtering heuristic still cannot filter out the noise from the transcription results of these pieces well. However, where actual notes and extraneous notes are separated well by the median of the overall pitch distribution, this makes low-pitch audio tend to be retrieved more easily.

The results for SBDG and SBDL show that it is possible to align a whole piece of audio music file with its symbolic equivalent representation by using dynamic programming techniques that had previously been shown effective for aligning symbolic sequences. However, the effectiveness is quite low, and as we shall see in Section VI-B, using pitch classes is more appropriate.

B. Pitch Classes Approach

As reported in our other work [34], representing the notes in both the transcribed audio and symbolic sequences as pitch classes supports effective retrieval. To retrieve audio files using their symbolic equivalent, the best approach is to use the LCS score divided by the square of the natural logarithm of the answer as the similarity measurement. Using the same collection and query set as we use in this paper, the MAP value was 0.826. The work also reports that using only the first 100 symbols for each query ($q_0, \ldots, q_{99}$) with $S_d(q, a, 1.1)$ supports highly effective retrieval (a MAP value of 0.768 is achieved). The $S_d(q, a, d)$ measure was tested with $d \in \{0.5, 0.6, 0.7, \ldots, 1.5, 2.0, 3.0, 4.0, 5.0\}$. The results for these values are reported in Suyoto *et al.* [34]. We use the optimal value of d in this paper.

However, users do not always know the first notes of a song. Sometimes, they only remember a part in the middle or the ending of the song. Considering this, we experiment further by varying the song parts used in the queries. In particular, we use the middle 100 notes ($q_{\lfloor |q|/2 \rfloor - 50}, \ldots, q_{\lfloor |q|/2 \rfloor + 49}$) and the last 100 notes ($q_{|q|-100} \ldots q_{|q|-1}$). Similarly to the previous experiment, if a query is shorter than 100 symbols, it remains untruncated. The mean precision at N curves are shown in Fig. 10. Table III shows the MAP values for those retrieval methods. Using the first 100 notes is overall better than the ending 100 notes (paired t-test, $p < 0.05$) but not overall better than the middle 100 notes (paired t-test, $p > 0.1$). There is no evidence that using the middle 100 notes is overall better than the ending 100 notes (paired t-test, $p > 0.1$).

However, if we look at mean precision at 10, our statistical tests reveal that varying the song parts for the queries does not cause significant difference in effectiveness. Using a paired t-test, the p value for all comparisons is greater than 0.05. This shows that we cannot draw the conclusions that using the beginning part of the songs as queries is consistently better than using other parts of the songs. Therefore, the result of this new

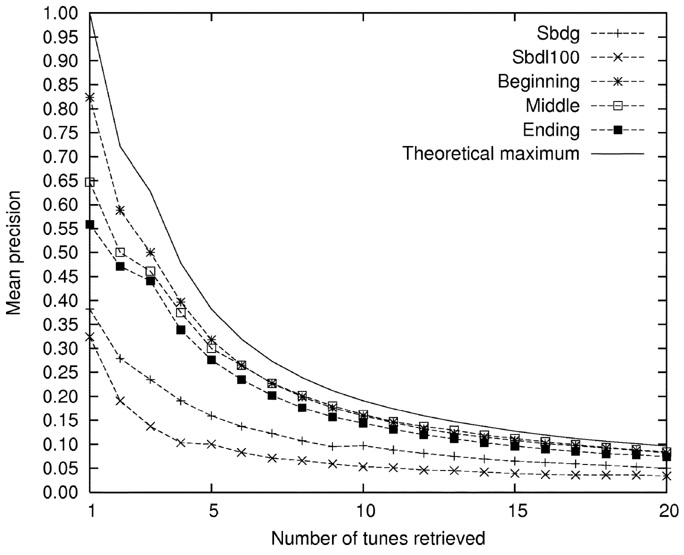

Fig. 10. Mean precision at N ($\langle P_N \rangle$) curves for SBDG, SBDL100, and $S_d(q', a, 1.1)$ using the beginning, middle, and ending notes of the full query.

TABLE III
MAP VALUES FOR SBDG, SBDL100, AND $S_d(q', a, 1.1)$ USING THE BEGINNING, MIDDLE, AND ENDING NOTES OF THE FULL QUERY. THE BEST VALUE IS HIGHLIGHTED

Method	MAP
SBDG	0.374
SBDL100	0.229
Beginning	0.768
Middle	0.651
Ending	0.566

approach reveals that users need not necessarily use the first few notes of a song to form effective queries.

Recall that in Section V we mention that for BWV 1042, the pieces in the collection are polyphonic whereas the queries as monophonic. Generally, our method performs poorly for these. There is an exception for using the middle part of BWV1042_AA, which successfully retrieves a correct answer in the first rank position. However, this seems to be coincidental and generally, our method fails for the monophonic cases. The retrieval effectiveness may be improved by setting an appropriate window size. This is still subject to further experimentations.

Compared to the results for the relative pitch methods (Section VI-A), the results obtained from using pitch classes show that this approach supports more effective retrieval. Further, by using pitch classes, the burden on users can be lowered since it is shown that $S_d(q, a, 1.1); |q| = 100$ performs more effectively than SBDG. Significance tests on our data reveal that using $S_d(q, a, 1.1)$ is significantly more effective than using any of the relative pitch methods on overall, based on the average precision values (paired t-test, $p < 0.05$). Two aspects affect the superiority of $S_d(q, a, d)$: alignment algorithm and representation. The relative pitch methods employ alignment algorithms with penalties, while the pitch classes methods employ the LCS algorithm, which is equivalent to both global and local alignment with weightless penalties. The amount of noise between matching symbols in the transcriptions is also a key factor: since it is relatively high it lowers the effectiveness of mismatch and insertion/deletion penalties. The same thing also happens when the noise removal and bass-part extraction stages eliminate matching symbols. At the same time, the amount of noise and eliminated matching symbols also causes the relative pitch representation to be inaccurate.

Recall that across our query set, the average query length is 814.15 symbols. With query lengths of 100 (only one query in our set is shorter than 100), i.e., only 12.3% of the average, effective retrieval is achievable. Therefore, our approach reduces the burden on users when creating queries substantially.

VII. Conclusion and Future Work

This paper proposes and evaluates the feasibility of using various approximate string matching approaches for retrieving audio by using manually constructed queries. Our results show the following.

- Effective retrieval of audio by using symbolic queries is possible, with MAP of 77%.
- Truncated queries are as effective as full-length ones. Moreover, they reduce the burden on users by allowing them to issue short queries.
- Truncated queries are effective no matter whether they are from the beginning, middle, or end of a piece.
- Representing notes as their pitch classes facilitates more effective retrieval than using a relative pitch standardization, because the former is not severely affected by the noise in transcriptions.

We plan to explore this technique on larger collections to investigate its scalability. We also plan to explore indexing techniques to increase the efficiency of retrieval. Once this is successful, we will be able to construct a practical content-based music retrieval system that accepts partial symbolic input such as segments of musical scores. Another step that can be taken is investigating the effectiveness of our approaches on transcriptions of various accuracy levels.

Acknowledgment

The authors would like to thank J. Zobel for obtaining TS-AudioToMidi and S. Garcia for his input on automating the transcription process. They would also like to thank the anonymous reviewers for their helpful comments.

References

[1] J.-J. Aucouturier and M. Sandler, J. S. Downie and D. Bainbridge, Eds., "Using long-term structure to retrieve music: Representation and matching," in *Proc. 2nd Int. Symp. Music Inf. Retrieval*, Bloomington, IN, Oct. 2001, pp. 1–2.

[2] J.-J. Aucouturier and M. Sandler, "Finding repeating patterns in acoustical musical signals: Applications for audio thumbnailing," in *Proc. Audio Eng. Soc. 22nd Int. Conf. Virtual, Synthetic, Entertainment Audio*, Espoo, Finland, Jun. 2002, pp. 412–421.

[3] L. P. Clarisse, J. P. Martens, M. Lesaffre, B. D. Baets, H. D. Meyer, and M. Leman, M. Fingerhut, Ed., "An auditory model based transcriber of singing sequences," in *Proc. 3rd Int. Conf. Music Inf. Retrieval*, Paris, France, Oct. 2002, pp. 116–123.

[4] R. B. Dannenberg, W. P. Birmingham, G. Tzanetakis, C. Meek, N. Hu, and B. Pardo, H. H. Hoos and D. Bainbridge, Eds., "The Musart testbed for query-by-humming evaluation," in *Proc. 4th Int. Conf. Music Inf. Retrieval*, Baltimore, MD, Oct. 2003, pp. 41–47.

[5] J. Eggink and G. J. Brown, C. L. Buyoli and R. Loureiro, Eds., "Extracting melody lines from complex audio," in *Proc. 5th Int. Conf. Music Inf. Retrieval*, Barcelona, Spain, Oct. 2004, pp. 84–91.

[6] J. Foote, D. Byrd, J. S. Downie, T. Crawford, W. B. Croft, and C. Nevill-Manning, Eds., "ARTHUR: Retrieving orchestral music by long-term structure," in *Proc. 1st Int. Symp. Music Inf. Retrieval*, Plymouth, MA, Oct. 2000.

[7] D. C. Giancoli, *Physics: Principles with Applications*, 6th ed. Upper Saddle River, NJ: Pearson Education, 2004.

[8] E. Gómez and P. Herrera, R. Dannenberg, K. Lemström, and A. Tindale, Eds., "The song remains the same: Identifying versions of the same piece using tonal descriptors," in *Proc. 7th Int. Conf. Music Inf. Retrieval*, Victoria, BC, Canada, Oct. 2006, pp. 180–185.

[9] A. Guo and H. Siegelmann, C. L. Buyoli and R. Loureiro, Eds., "Time-warped longest common subsequence algorithm for music retrieval," in *Proc. 5th Int. Conf. Music Inf. Retrieval*, Barcelona, Spain, Oct. 2004, pp. 258–261.

[10] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge, U.K.: Cambridge Univ. Press, 1997.

[11] N. Hu and R. B. Dannenberg, G. Marchionini and W. Hersh, Eds., "A comparison of melodic database retrieval techniques using sung queries," in *Proc. 2nd ACM/IEEE-CS Joint Conf. Digit. Libraries*, Portland, OR, Jul. 2002, pp. 301–307.

[12] N. Hu, R. B. Dannenberg, and G. Tzanetakis, "Polyphonic audio matching and alignment for music retrieval," in *Proc. 2003 IEEE Workshop Appl. Signal Process. Audio Acoust.*, New Paltz, NY, Oct. 2003, pp. 185–188.

[13] J.-S. R. Jang, C.-L Hsu, and H.-R. Lee, J. D. Reiss and G. A. Wiggins, Eds., "Continuous HMM and its enhancement for singing/humming query retrieval," in *Proc. 6th Int. Conf. Music Inf. Retrieval*, London, U.K., Sep. 2005, pp. 546–551.

[14] A. Klapuri, R. Dannenberg, K. Lemström, and A. Tindale, Eds., "Multiple fundamental frequency estimation by summing harmonic amplitudes," in *Proc. 7th Int. Conf. Music Inf. Retrieval*, Victoria, BC, Canada, Oct. 2006, pp. 216–221.

[15] M. Marolt, C. L. Buyoli and R. Loureiro, Eds., "Gaussian mixture models for extraction of melodic lines from audio recordings," in *Proc. 5th Int. Conf. Music Inf. Retrieval*, Barcelona, Spain, Oct. 2004, pp. 80–83.

[16] M. Marolt, R. Dannenberg, K. Lemström, and A. Tindale, Eds., "A mid-level melody-based representation for calculating audio similarity," in *Proc. 7th Int. Conf. Music Inf. Retrieval*, Victoria, BC, Canada, Oct. 2006, pp. 280–285.

[17] D. Mazzoni and R. B. Dannenberg, "Melody matching directly from audio," in *Proc. 2nd Int. Symp. Music Inf. Retrieval*, J. S. Downie and D. Bainbridge, Eds., Bloomington, IN, Oct. 2001, pp. 17–18.

[18] M. Müller, F. Kurth, and M. Clausen, J. D. Reiss and G. A. Wiggins, Eds., "Audio matching via chroma-based statistical features," in *Proc. 6th Int. Conf. Music Inf. Retrieval*, London, U.K., Sep. 2005, pp. 288–295.

[19] M. Müller, F. Kurth, and T. Röder, C. L. Buyoli and R. Loureiro, Eds., "Towards an efficient algorithm for automatic score-to-audio synchronization," in *Proc. 5th Int. Conf. Music Inf. Retrieval*, Barcelona, Spain, Oct. 2004, pp. 365–372.

[20] R. P. Paiva, T. Mendes, and A. Cardoso, J. D. Reiss and G. A. Wiggins, Eds., "On the detection of melody notes in polyphonic audio," in *Proc. 6th Int. Conf. Music Inf. Retrieval*, London, U.K., Sep. 2005, pp. 175–182.

[21] B. Pardo and M. Sanghi, J. D. Reiss and G. A. Wiggins, Eds., "Polyphonic musical sequence alignment for database search," in *Proc. 6th Int. Conf. Music Inf. Retrieval*, London, U.K., Sep. 2005, pp. 215–222.

[22] J. Pickens, J. P. Bello, G. Monti, T. Crawford, M. Dovey, M. Sandler, and D. Byrd, M. Fingerhut, Ed., "Polyphonic score retrieval using polyphonic audio queries: A harmonic modelling approach," in *Proc. 3rd Int. Conf. Music Inf. Retrieval*, Paris, France, Oct. 2002, pp. 140–149.

[23] J. Pickens and C. Iliopoulos, J. D. Reiss and G. A. Wiggins, Eds., "Markov random fields and maximum entropy modeling for music information retrieval," in *Proc. 6th Int. Conf. Music Inf. Retrieval*, London, U.K., Sep. 2005, pp. 207–214.

[24] A. Pikrakis and S. Theodoridis, J. D. Reiss and G. A. Wiggins, Eds., "A novel HMM approach to melody spotting in raw audio recordings," in *Proc. 6th Int. Conf. Music Inf. Retrieval*, London, U.K., Sep. 2005, pp. 652–657.

[25] G. E. Poliner and D. P. W. Ellis, J. D. Reiss and G. A. Wiggins, Eds., "A classification approach to melody transcription," in *Proc. 6th Int. Conf. Music Inf. Retrieval*, London, U.K., Sep. 2005, pp. 161–166.

[26] C. Raphael, C. L. Buyoli and R. Loureiro, Eds., "A hybrid graphical model for aligning polyphonic audio with musical scores," in *Proc. 5th Int. Conf. Music Inf. Retrieval*, Barcelona, Spain, Oct. 2004, pp. 387–394.

[27] M. Sanderson and J. Zobel, R. A. Baeza-Yates, N. Ziviani, G. Marchionini, A. Moffat, and J. Tait, Eds., "Information retrieval system evaluation: Effort, sensitivity, and reliability," in *Proc. 28th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Salvador, Brazil, Aug. 2005, pp. 162–169.

[28] S. Shalev-Shwartz, S. Dubnov, N. Friedman, and Y. Singer, K. Järvelin, M. Beaulieu, R. Baeza-Yates, and S. H. Myaeng, Eds., "Robust temporal and spectral modeling for query by melody," in *Proc. 25th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Tampere, Finland, Aug. 2002, pp. 331–338.

[29] S. Shalev-Shwartz, J. Keshet, and Y. Singer, C. L. Buyoli and R. Loureiro, Eds., "Learning to align polyphonic music," in *Proc. 5th Int. Conf. Music Inf. Retrieval*, Barcelona, Spain, Oct. 2004, pp. 381–386.

[30] J. Shifrin and W. P. Birmingham, H. H. Hoos and D. Bainbridge, Eds., "Effectiveness of HMM-based retrieval on large databases," in *Proc. 4th Int. Conf. Music Inf. Retrieval*, Baltimore, MD, Oct. 2003, pp. 33–39.

[31] J. Song, S. Y. Bae, and K. Yoon, M. Fingerhut, Ed., "Mid-level music melody representation of polyphonic audio for query-by-humming system," in *Proc. 3rd Int. Conf. Music Inf. Retrieval*, Paris, France, Oct. 2002, pp. 133–139.

[32] F. Soulez, X. Rodet, and D. Schwarz, H. H. Hoos and D. Bainbridge, Eds., "Improving polyphonic and poly-instrumental music to score alignment," in *Proc. 4th Int. Conf. Music Inf. Retrieval*, Baltimore, MD, Oct. 2003, pp. 143–148.

[33] I. S. H. Suyoto and A. L. Uitdenbogerd, L. Zhou, B. C. Ooi, and X. Meng, Eds., "Effectiveness of note duration information for music retrieval," in *Proc. 10th Int. Conf. Database Syst. Adv. Applicat.*, Beijing, China, Apr. 2005, LNCS 3453, pp. 265–275.

[34] I. S. H. Suyoto, A. L. Uitdenbogerd, and F. Scholer, "Effective retrieval of polyphonic audio with polyphonic symbolic queries," in *Proc. 9th ACM SIGMM Int. Workshop Multimedia Inf. Retrieval*, Sep. 2007, pp. 105–114.

[35] R. Typke, F. Wiering, and R. C. Veltkamp, C. L. Buyoli and R. Loureiro, Eds., "A search method for notated polyphonic music with pitch and tempo fluctuations," in *Proc. 5th Int. Conf. Music Inf. Retrieval*, Barcelona, Spain, Oct. 2004, pp. 281–288.

[36] A. L. Uitdenbogerd, "Music information retrieval technology," Ph.D. dissertation, School Comput. Sci. Inf. Technol., Royal Melbourne Inst. Technol., Melbourne, Australia, 2002.

[37] A. L. Uitdenbogerd and J. Zobel, D. Bulterman, K. Jeffay, and H. J. Zhang, Eds., "Melodic matching techniques for large music databases," in *Proc. 1999 ACM Multimedia Conf.*, Orlando, FL, Nov. 1999, pp. 57–66.

[38] E. M. Voorhees and C. Buckley, K. Järvelin, M. Beaulieu, R. Baeza-Yates, and S. H. Myaeng, Eds., "The effect of topic set size on retrieval experiment error," in *Proc. 25th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Tampere, Finland, Aug. 2002, pp. 316–323.

[39] A. Wang, "The Shazam music recognition service," *Commun. ACM*, vol. 49, no. 8, pp. 44–48, Aug. 2006.

Iman S. H. Suyoto received the Sarjana Sains degree in physics from the University of Indonesia, Depok, Indonesia, in 2001 and the M.Tech. degree in information technology from the Royal Melbourne Institute of Technology (RMIT), Melbourne, Australia, in 2003. He is currently pursuing the Ph.D. degree in the School of Computer Science and Information Technology, RMIT.

He currently works as a sessional academic at the School of Computer Science and Information Technology, RMIT. His research interests include music information retrieval, approximate string matching, and computer science education.

Mr. Suyoto is a member of the RMIT Search Engine Group and ACM.

Alexandra L. Uitdenbogerd received the Grad.Dip. degree in education from the University of Melbourne, Melbourne, Australia in 1992, the B.Sc. degree in computer science from University of Western Australia, Perth, in 1986, and the Ph.D. degree in computer science from the Royal Melbourne Institute of Technology (RMIT), Melbourne, in 2002.

She is a Senior Lecturer at the School of Computer Science and Information Technology, RMIT. Her research interests include music information retrieval, recommender systems, text retrieval on linguistic criteria, pattern matching, compression, and computer-assisted language learning.

Dr. Uitdenbogerd is a member of the RMIT Search Engine Group, the Australian Computer Music Association, and the Melbourne Composers League. She has been a Program Committee member for ACM SIGIR, ISMIR, and the Australasian Computer Science Conference.

Falk Scholer received the M.Sc. degree in computer science from University College London, London, U.K., in 1996 and the Ph.D. degree in computer science from the Royal Melbourne Institute of Technology (RMIT), Melbourne, Australia, in 2004.

He currently works as a Lecturer at the School of Computer Science and Information Technology, RMIT, and conducts research in the area of information retrieval. His research interests include the evaluation of search systems, interactive information retrieval, past queries and query log analysis, music retrieval, and efficient indexing.

Dr. Scholer is a member of the RMIT Search Engine Group and ACM SIGIR.