Use of Locality Sensitive Hashing in Query by Humming Method of Audio Signal

Sonal Goel
Manav Rachna College of Engineering
Faridabad
E-mail: sonalgoel23@gmail.com

Mehak Malik
Manav Rachna College of Engineering
Faridabad

Charu Pathak
Department of Electronics & Communication
Manav Rachna University, Faridabad
E-mail: charuup123@gmail.com

Abstract: This paper is based on Locality Sensitive Hashing (LSH) which proposes a query by humming method. This method constructs an index of melodic fragments by extracting pitch vectors. This method automatically deciphers a song query into notes and then concentrates the pitch vectors as similar to the index construction. This method searches for similar fragments in the database to obtain a directory of candidate melodies for each query pitch vector. This is performed efficiently by using LSH. In our experiments, the method achieved mean reciprocal rank of 0.885 for 2797 queries when searching from database of 6030 Musical Instrumental Digital Interface (MIDI) melodies. MIDI allows multiple instruments to be played from a single controller, which makes stage setups much more portable.

Keywords: *Music, Information retrieval, Database query processing, Audio Systems*

I. INTRODUCTION

Query by humming (QBH) refers to music information retrieval systems where short audio clips of singing or humming act as queries. If the user does not remember the name of the artist or the song to make a metadata query, a natural option is to sing, hum, or whistle a part of the melody of the song into a microphone and let a QBH system to retrieve the song. The QBH task can be broadly divided into two subparts:- i) converting a query into a format which enables robust searching and ii) matching the query with melodies in the database. The former problem is often associated with automatic transcription of a query into temporally segmented note events or into frame-wise measured pitch trajectory, whereas the latter concentrates on measuring melodic similarity. The former problem is often associated with automatic transcription of a query into temporally segmented note events or into frame-wise measured pitch trajectory, whereas the latter concentrates on measuring melodic similarity.

The major challenges for QBH systems include i) handling of highly varying quality of queries, ii) huge size of melody databases, and iii) automatic production of the melody databases. First, the quality of queries may vary drastically in terms of staying in tune and tempo and also in the recording quality of the query audio. Second, linear search over database items is not acceptable due to huge databases of music. Third, most of the QBH research has concentrated on searching from databases of MIDI melodies and it would be highly desirable to obtain such databases directly from music recordings. From an application point of view, this would also enable immediate playback of the retrieved melody segments in the original music piece.

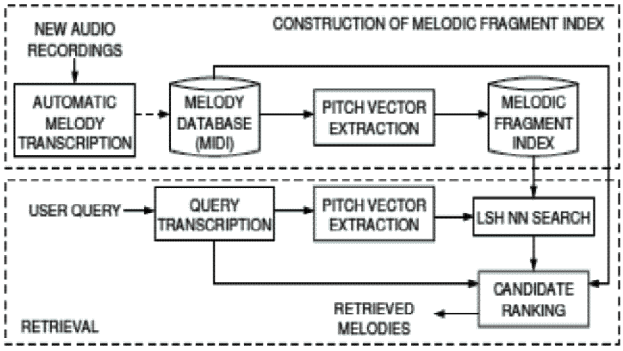

Fig. 1. Block diagram of Proposed model

Fig. 1 shows a block diagram of the method. Given a database of melodies in MIDI format, the method constructs an index of melodic fragments by extracting pitch vectors. A pitch vector stores an approximate representation of melody contour within a fixed-length time window. In retrieval, the method automatically converts a query into MIDI notes and then extracts pitch vectors. For each query pitch vector, the method

searches for nearest neighbors in Euclidean space from the index of database melody fragments to obtain melody candidates and their matching positions in time. This can be performed very efficiently by using locality sensitive hashing (LSH). Final ranking of candidates is done by comparing the whole transcribed query to each candidate melody segment. Due to the melodic fragment index, the method manages long database melodies directly, without having to segment melodies into phrases. Also, the queries do not have to start from the beginning of a melodic phrase. Using LSH provides a significant speed-up and retrieval performance comparable to the state-of-the-art.

II. LITERATURE REVIEW

Apart from the clinical studies, the human voice is also studied in the contexts of singing and speech. In singing, the sounds is produced by an airstream from the lungs outwards, which then passes through the glottis the vocal tract. These vibrations of vocal cords produce the fundamental frequency and its harmonic partials thus producing the music. The vocal tract acts as an amplifier thus increasing and decreasing the amplitude of each partial [13]. The mechanisms of producing a sound by means of mouth and vocal tract exceed in number and variety those involved in context such as percussive sounds, by clashing teeth, rarely found in speech. There have been several works to partially in sound retrieval using vocalization. The evolution in this area by a number of specific techniques as Query by Humming has been summarized [14, 15, and 16].

The sound synthesis also becomes better by the analysis of a vocal signal. The pitch and loudness are the most commonly used factors used for information retrieval of the vocal signal.

III. CONSTRUCTON OF INDEX OF MELODIC FRAGMENTS

The method constructs an index which stores melodic fragments, their temporal positions within the database melodies, and melody identifiers. The melody identifier determines the song from which a melody fragment has been extracted. The index enables efficient retrieval of melodies from the database.

A. PITCH VECTOR EXTRACTION

A melody is here defined as a sequence of L notes n1:L, where i:th note ni = _pi, bi, ei_ is defined by pitch pi in MIDI note numbers, and the onset time bi and the offset time ei of the note in seconds. The melodic fragments are represented as pitch vectors which are extracted from such note sequences. Given a melody n1:L, the pitch vectors are extracted as follows. First, rests between consecutive notes are removed by extending the offset of a preceding note to the onset of the following note, i.e., ei $\leftarrow$ bi+1 for i = 1, . . . , L " 1. Then for each note i, a pitch vector pi of length d is extracted by determining the melody pitch values within w-second window starting from the note onset bi. Fig. 2 shows an example of pitch vector extraction by using window size w = 3 s and vector length d = 20. The top panel shows a part of a melody and the bottom panels show the first four pitch vectors pi and their extraction positions bi.

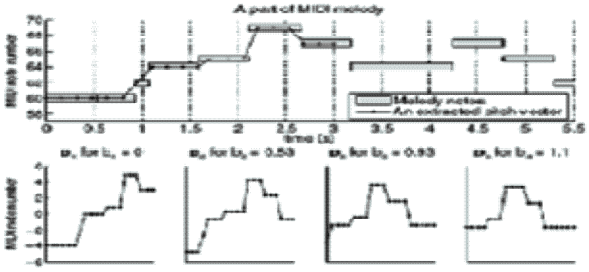

Fig. 2. An example of pitch vector extraction

B. SIMILARITY OF MELODY FRAGMENTS

The similarity of melodic fragments is here measured using Euclidean distance between pitch vectors. Formally, two pitch vectors **pi** and **pj** define points in d-dimensional space where the distance is given by-

$$\|pi - pj\| = \left(\sum_{k=1}^{d} |pi(k) - pj(k)|^2 \right)^{(1/2)} \quad \ldots (1)$$

Given a melodic fragment defined by point pi, we can find similar fragments in the index by searching for nearest neighbors (NNs) of the point, i.e., all the points to which the distance is less than a specified threshold r. This could be done by simply measuring the distance of pi to all the vectors in the database. However, this results in a search time that depends linearly on the database size. To obtain a sub linear time complexity,

we use locality sensitive hashing [8, 9]. LSH is a randomized algorithm for searching approximately nearest neighbors in high dimension spaces.

III. QUERY PROCESSING AND RETRIEVAL OF MELODY

The retrieval stage consists of the following steps: i) transcription of a sung query into notes, ii) extraction of pitch vectors from the notes, iii) retrieving similar melodic fragments in the database using LSH, and iv) performing final ranking of the retrieved melody candidates.

C. QUERY TRANSCRIPTION AND TUNING

A sung query is first converted into a note sequence. For this task, we use a melody transcription method designed for polyphonic music. Although it is not necessary to handle polyphony in query transcription, this method is used also to produce a melody database directly from music recordings (see Fig. 1). The method is an improved version of [12]. Briefly, the method uses a frame-wise pitch salience estimator to measure the strength of different fundamental frequencies in 92.9 ms analysis frames with 23.2 ms interval between successive frames. This feature extractor is followed by HMMs representing melody notes and the background. The method also applies a musicological model to control between-note transitions. As an output, the method produces a sequence of notes in the format introduced in Sec. 2. One could also use some other melody transcription method which produces note sequences, e.g., the method proposed in [12]. The queries are not likely performed in absolute tuning (MIDI note 69 at 440 Hz), i.e., the tuning of a sung note can be between two integer MIDI pitches. Therefore, each transcribed query note is tuned by shifting it in frequency at most half a semitone up or down so as to maximize pitch salience within the note. Fig. 3 shows an example of transcribed and tuned query notes with the estimated pitch silences..

D. RETRIEVAL OF MELODIES

The tuned query note sequence is then used to retrieve similar melodic fragments from the database by extracting pitch vectors from the query. Let mj denote j:th window size modifier. For each query note onset bq, the method samples M pitch vectors using

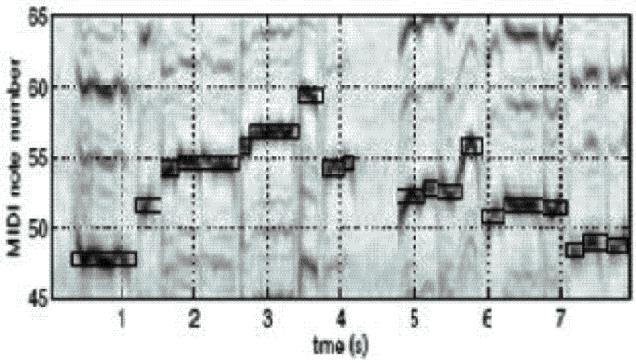

Fig. 3. A transcribed query. The grey-level intensity indicates the estimated pitch salience and the black boxes show the transcribed and tuned query notes

window sizes wmj , j = 1, . . ., M. A modifier value m < 1 implies that a query melodic fragment is performed faster than the database melodic fragment. Respectively, a slower performance is indicated by modifier values greater than one.

A reasonable range for window modifier values is between 0.65–1.7. For each query pitch vector, the method then searches for similar melodic fragments in the database using LSH. The LSH returns the nearest neighbours and their distances to the query point as matches. After retrieving matches for all the query points, we have a list of candidate melodies for final ranking

E. FINAL RANKING

To obtain the final list of retrieved melodies, the candidate melodies are ranked according to their distance to the entire query note sequence. The ranking is performed by examining all the matches preserved in the previous step. One match is denoted by _bq, mj, bc, s_, where bq is the extraction position of the query pitch vector, mj is the used window size modifier, bc is the extraction position of the database melodic fragment, and s is the song identifier. In addition, let t(q)0 and t(q)1 denote the onset time of the first query note and the offset time of the last query note, respectively. Then the time region corresponding to the entire query in the candidate melody is defined by t(c)0 = bc”(bq”t(q)0)/ mj and t(c) 1 = bc+(t(q)1 “bq)/mj . Hereafter, the term candidate segment refers to this time region within the candidate melody. The upper panel in Fig. 4 shows an example how the candidate segment is determined for one match. The entire query and the candidate segment

are then normalized both in pitch and time for distance calculation.

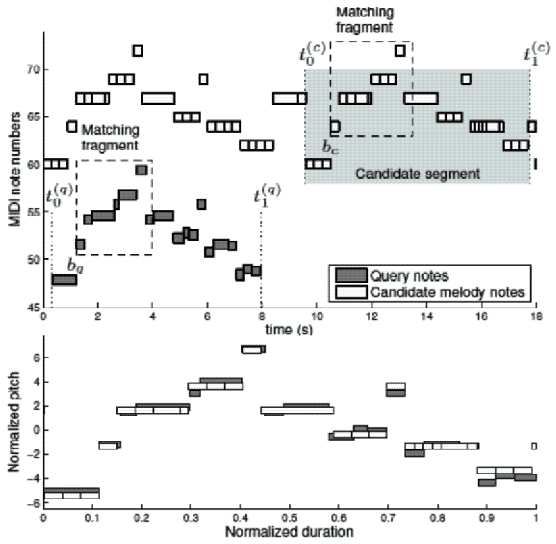

Fig. 4. A query and a matching candidate melody. The upper panel shows the matching fragments in these melodies and the determined candidate segment (light grey area). The lower panel shows the normalized query and the normalized matching segment for distance evaluation

The lower panel in Fig. 4 shows the normalized query and the normalized candidate segment. The distance between the normalized query and the normalized candidate segment is evaluated by using recursive alignment (RA) proposed by Wu et al. [7]. The above distance evaluation is performed for each match. The minimum distance match per candidate melody is preserved, since there may exist several candidate segments per melody. Finally, the list of candidates is sorted in ascending distance order and returned to the user.

IV. RESULTS

The method performance is measured using mean reciprocal rank (MRR) and top-X hit rate criteria.

For the Jang's corpus, the method reached MRR of 0.885 and top-3 hit rate of 90%. For these results, it was sufficient to preserve only two smallest-distance matches per query point which returned on the average 134 candidate melodies for a query. Interestingly, MRR of 0.592 could be reached just by ranking the candidate melodies according to the number of matching melodic fragments.

Table 1: Melody Retrieval Results

			MRR	Top-X hit rate (%)				
Corpus	*N*	*Dx*		1	3	5	10	20
Jang	2797 2797	6030 2048	0.885 0.909	86 89	90 92	91 93	92 94	93 95
Music	159	427	0.578	52	58	62	69	73

The used number M of window size modifiers mj was 17, and recursive alignment was used with five possible division points and two recursion levels.

With a Matlab implementation running on a 3.2 GHz Pentium 4 processor, the mean query time was 4.5 seconds of which the query transcription takes approximately 30%, LSH 32%, and the final ranking 13%. LSH provided a speed-up of factor 4–20 compared to exact nearest neighbour search in candidate retrieval without losing any accuracy in results. Retrieval errors are mostly related to the query performances: some queries differ from the correct answer so much that matching is very challenging even for human.

The results for the automatically transcribed melody database are expectedly worse than with the manually prepared MIDI files in Jang's corpus. However, the method achieved MRR of 0.578 which is rather encouraging result and motivates for further study.

V. CONCLUSIONS

A QBH method based on LSH and achieved retrieval performance comparable to the state-of-the-art has been presented in this paper. The method also achieved promising results for QBH of audio. The current representation of melodic fragments enables accurate results but contains redundant information and consequently uses memory inefficiently.

Future work includes development of a more compact representation. In addition, retrieval directly from music seems very promising for future development.

VI. REFERENCES

[1] R. Typke, Music Retrieval based on Melodic Similarity, Ph.D. thesis, Universiteit Utrecht, 2007.

[2] K. Lemstr¨om, String Matching Techniques forMusic Retrieval, Ph.D. thesis, University of Helsinki, 2000.

[3] C. Meek and W. Birmingham, "Applications of binary classification and adaptive boosting to the query-by-humming problem," in Proc. 3rd International Conference on Music InformatioRetrieval, 2002.

[4] J.-S. R. Jang, C.-L. Hsu, and H.-R. Lee, "Continuous HMM and its enhancement for singing/humming query retrieval," in Proc. 6th International Conference on Music Information Retrieval, 2005.

[5] J.-S. R. Jang and M.-Y. Gao, "A query-by-singing system based on dynamic programming," in Proc. International Workshop on Intelligent Systems Resolutions, 2000.

[6] A. Duda, A. N¨urnberger, and S. Stober, "Towards query byN humming/singing on audio databases," in Proc. 7th International Conference on Music Information Retrieval, 2007.

[7] X. Wu, M. Li, J. Yang, and Y. Yan, "A top-down approach to melody match in pitch countour for query by humming," in Proc. International Conference of Chinese Spoken Language Processing, 2006.

[8] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), 2006, pp. 459–468.

[9] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, "Locality sensitive hashing scheme based on p-stable distributions," in Proc. ACM Symposium on Computational Geometry, 2004, pp. 253–262.

[10] M. Casey and M. Slaney, "Fast recognition of remixed music audio," in Proc. 2007 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2007.

[11] M. Covell and S. Baluja, "Known-audio detection using Waveprint: Spectrogram fingerprinting by wavelet hashing," in Proc. 2007 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2007.

[12] M. Ryyn¨anen and A. Klapuri, "Transcription of the singing melody in polyphonic music," in Proc. 7th International Conference on Music Information Retrieval, 2006.

[13] Sundberg J (1977) The Acoustics of the Singing Voice, Scientific American offprints, vol 356. W.H Freeman, New York Ghias A, Logan J, Chamberlin D, Smith BC (1995)

[14] Query by humming: Musical information retrieval in an audio database. In: Proceedings of the Third ACM International Conference on Mul-timedia, MULTIMEDIA '95. ACM, New York, pp 231–236

[15] Nagavi TC, Bhajantri NU (2012) An extensive analysis of query by singing/humming system through query proportion. Int J Multimed Appl 4(6). doi:10.5121/ijma.2012.4606

[16] Weinstein E (2005) Query by humming: a survey. Tech. rep., NYU and Google

❒