Audio Engineering Society

Convention Paper

Presented at the 120th Convention
2006 May 20–23 Paris, France

Evaluation of Query-by-Humming Systems using a Random Melody Database

Jan-Mark Batke[1] and Gunnar Eisenberg[2]

[1]*THOMSON, Corporate Research, Audio Processing Lab, D-30625 Hannover, Germany*

[2]*Communications Systems Group, Technische Universität Berlin, Germany*

Correspondence should be addressed to Jan-Mark Batke (Jan-Mark.Batke@thomson.net)

ABSTRACT

The performance of melody retrieval using a query-by-humming (QBH) system depends on different parameters. For the query, parameters like length of the query and possibly contained errors influence the success of the retrieval. But also the size of the melody database inside the QBH-system has a certain impact on the query. This paper describes how the statistical parameters of a random melody database are modelled to get the same behaviour like a database containing authentic melodies. Databases containing random melodies are a testing facility to QBH-systems.

1. INTRODUCTION

A query-by-humming (QBH) system enables a user to hum a melody as a query to retrieve a list of songs with matching melodies. Generally speaking, such systems belong to the family of music information retrieval (MIR) systems. Most QBH-systems process the query in two stages. First, the transcription stage converts the acoustic input of the user to a symbolic representation of the melody [1–3]. Second, this extracted melody representation is compared to the content of a melody database (MDB). A list of matching songs is the result of this comparison stage.

A commonly used symbolic representation for melody descriptions inside a QBH-system is the *melody contour*. Unlike symbolic music representations like MIDI (music instrument digital interface) the melody contour describes only whether the melody's pitch goes up, down or remains the same. The popular Parsons code is providing this information using only three symbols, »u« for up, »d« for down and »r« for repeated pitch (also »s« for same) [4]. A more detailed contour description is given by the *interval contour* [5]. It is using the number of semitones the melody is changing note for note. In this paper, a five step contour representation is used that extends the Parsons Code, using »U« and »D« for large intervals of a minor third and more up and down, respectively. Note that also the MPEG-7 melody contour provides such a five step representation [6].

The comparison of melody contours is carried out using a similarity measure. Since the melody contour representation can be seen as a chain of symbols, e. g. a string, matching algorithms like *longest common subsequence*, *longest common substring* or *local alignment* can be used [5]. Another popular approach is to use n-gram methods like the *Ukkonen measure*, *sum of frequencies* or *local alignment* [5, 7, 8]. All these methods provide a numerical value of similarity which enables the comparison stage to determine how similar the user's query is to each melody in the MDB.

The performance of a QBH-system depends obviously on the functionality of the transcription and the comparison stage. Furthermore, the user's errors like missing notes (deletions), additionally notes (inserts) or just wrong notes (editions) contained in the query do influence the quality of retrieval [3, 5, 9, 10]. However, also the content of the MDB has a certain impact on the success of the query. Clearly the size, e. g. the number of entries in the MDB influences the result. The more melodies are in the MDB, the more melodies might be similar to the query and the correct melody is found more difficultly.

In this paper, the influence of the MDB content is evaluated. Usually the MDB is build from MIDI files. Selecting the melody from MIDI files however is a time consuming task. To get a large scaled MDB, in our experiments we measure the statistics of an existing MDB containing melodies extracted from MIDI data to model a new MDB. This MDB is filled with random melodies. Doing so an MDB of arbitrary size can be generated. It turns out that most important parameters of such a random MDB are the distribution of the melodies' length and the distribution of symbols used for melody description.

For convenience, all distance metrics used for the evaluation are briefly described in Section 2. The following Section 3 discusses the properties of MDBs. The evaluation of different MDBs containing random melodies is described in Section 4. Conclusions are drawn in Section 5.

2. DISTANCE METRICS

Distance metrics are used to determine the similarity of two melodies in a QBH-system. Their properties have a strong impact on the result of the comparison, of course. Several distance metrics are evaluated in this paper. Our attention is turned to their usability for the comparison of melody contours. The distance of two melodies describes their similarity. Different techniques like dynamic programming or n-gram methods can be employed for calculation of these similarities. The distance metrics described here are subdivided into these two families.

2.1. Dynamic Programming

String matching based on dynamic programming in its most basic form involves creating a two dimensional matrix **A** that stores the results of comparisons between two strings [5]. The similarity measure is the maximum value of this matrix, e. g.

$$R_{\mathrm{DP}} = \max \mathbf{A}. \tag{1}$$

2.1.1. Longest common subsequence

With l*ongest c*ommon sub*sequence* (LCE) technique, the query is matched against pieces with no penalty for gaps of any size between the matching symbols [5].

Let **A** be the two dimensional matrix, **q** is a vector with contour values of the query and **p** is the vector with the melody's contour values. Index i is running from 0 to length of the query and j is running from 0 to length of piece. The elements of the matrix **A** are calculated using

$$A[i,j] = \max \begin{cases} A[i-1,j] & i \geq 1 \\ A[i,j-1] & j \geq 1 \\ A[i-1,j-1]+1 & q(i) = p(j)\,; i,j \geq 1 \\ 0 & \end{cases} \tag{2}$$

The elements $A[i,j]$ are incremented in case of a match, else wise the value of the left upper diagonal is taken.

2.1.2. Local alignment

L*ocal* al*ignment* (LAL) can be modified by cost parameters for inserts, deletions, and editions. The result is never negative.

$$A[i,j] = \max \begin{cases} A[i-1,j]+d & i \geq 1 \\ A[i,j-1]+d & j \geq 1 \\ A[i-1,j-1]+e & q(i) = p(j)\,; i,j \geq 1 \\ A[i-1,j-1]+m & q(i) \neq p(j) \\ 0 & \end{cases} \tag{3}$$

The cost parameters are usually chosen to $d = 2$, $e = 1$ and $m = -1$. Matrix **A** shows how well sequence **q** can aligned to sequence **p**. Since the alignment is maximised locally, a maximum in the matrix indicates the best matching portion of the sequences.

2.1.3. Longest common substring

It is also meaningful to reset the result in $A[i,j]$ to zero if the symbols compared do not match.

$$A[i,j] = \max \begin{cases} A[i-1,j]+d & i \geq 1 \\ A[i,j-1]+d & j \geq 1 \\ A[i-1,j-1]+e & q(i) = p(j); i,j \geq 1 \\ 0 & q(i) \neq p(j) \end{cases} \quad (4)$$

The maximum in the matrix represents the length of the l*ongest* c*ommon* sub*string* (LCT).

2.2. N-gram methods

Melody strings can be broken down into n-grams, or substrings of a given length n for matching [5]. N-gram methods involve counting the common (or different) n-grams of the query and melody to arrive at a score representing their similarity. A melody contour described by I interval values is given by

$$\mathbf{c} = [m(1), m(2), \ldots, m(I)] \quad (5)$$

To create an n-gram of length N we build vectors

$$\mathbf{g}_N(i) = [m(i), m(i+1), \ldots, m(i+N-1)], \quad (6)$$

containing N consecutive interval values, where $i = 1 \ldots I-N+1$. The total amount of all n-grams is $I-N+1$. Again, $\mathbf{q}$ represents the vector with contour values of the query, and $\mathbf{p}$ is the piece to match against. Let Q_N and P_N be the sets of n-grams contained in $\mathbf{q}$ and $\mathbf{p}$, respectively.

2.2.1. Coordinate matching

Coordinate matching (CM), also known as *count distinct measure*, counts n-grams $\mathbf{g}_N(i)$, that commonly occur in $\mathbf{q}$ and in $\mathbf{p}$:

$$R_{\mathrm{CM}} = \sum_{\mathbf{g}_N(i) \in Q_N \cap P_N} 1. \quad (7)$$

2.2.2. Sum of frequencies

Sum of frequencies (SF) counts how often a n-gram $\mathbf{g}_N(i)$ that occurs both in $\mathbf{q}$ and $\mathbf{p}$ is contained in $\mathbf{p}$.

$$R_{\mathrm{SF}} = \sum_{\mathbf{g}_N(i) \in Q_N \cap P_N} U(\mathbf{g}_N(i), P_N) \quad (8)$$

All frequencies $U(\mathbf{g}_N(i), P_N)$ are summed up.

2.2.3. Ukkonen measure

The *Ukkonen measure* (UK) counts the n-grams that do *not* commonly occur in $\mathbf{p}$ and $\mathbf{q}$.

$$R_{\mathrm{UK}} = -\sum_{\mathbf{g}_N(i) \in Q_N \cup P_N} |U(\mathbf{g}_N(i), Q_N) - U(\mathbf{g}_N(i), P_N)| \quad (9)$$

It is sufficient to explore the set union $Q_N \cup P_N$ since only these n-grams contribute to the result. Large values indicate a big difference between the two sequences, therefore a minus sign is added to let the maximum value indicate best accordance.

2.3. Normalisation

For melody comparison in QBH-systems it is reasonable to normalise the distance metrics to the length of the melodies [5, 6, 11]. The normalisation can be varied using the melody's length, nth root of length or logarithm of length. According to results in [11] using a n-gram length of 6, UK and SF perform best using length, CM requires second root and LAL, LCE, and LCT require ninth root.

3. MELODY DATABASE

To describe a melody database (MDB) in general different statistical parameters can be employed. Besides of the size of the database, these are the distribution of melody lengths and melody contour symbols. Moreover, looking at the melody as a random process the conditional probabilities of symbols can be taken into account.

The MDB evaluated in this paper comprises 405 MIDI files retrieved from the Internet. The genre is mostly popular music with some additional classical tunes. The melodies are extracted using the melody track of the MIDI file, which is converted to a five step melody contour representation following the MPEG-7 standard [6].

3.1. Database size

In the literature very different sizes for MDBs are evaluated for use in QBH-systems. Small sized databases contain 50 titles like in [12], 100 titles like in [13] or 183 titles [1]. Medium sized databases contain about 400–1000 titles [6, 14–18]. There are some evaluations with large scaled databases like [19] using 3000 titles, [4, 20, 21] using about 10,000 titles and [22] using 39,925 titles. DANNENBERG et al. use a small collection of titles (258 songs

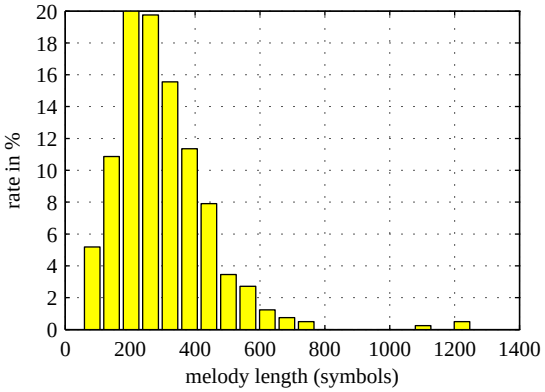


Figure 1: The histogram of melody lengths of the melody database used in this paper.

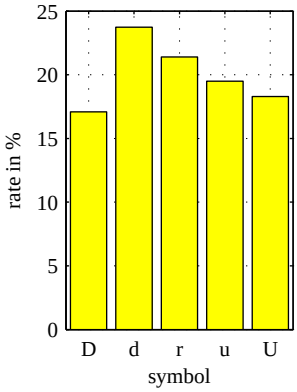


Figure 2: The distribution of symbols in the melody database.

of the Beatles) and extract single themes to get a larger database (resulting in 2844 themes in this case) [23].

On the other hand, existing databases of music download services contain about 2 million titles like iTunes [24], up to 6 million found at mp3.com [25]. The performance of QBH-systems using large scaled MDBs is therefore a point of interest.

3.2. Melody length

In this paper, the length of the melody denotes the total number of symbols. Talking about the melody of a tune the musical *theme* and the *leading voice* in a piece must be distinguished. In the latter case also all repetitions contribute to the melody's length. Information about the melody lengths distribution can be illustrated using a histogram and/or the mean length. KOSUGI et al. use a database holding about 10,000 titles and provide a histogram, the mean value for a the melody length is 365.16 symbols. DANNENBERG et al. extract singular themes as described in Section 3.1. The mean length is 41 symbols [23].

The distribution of lengths for the MIDI data used in this paper is depicted in Figure 1. The shape of the distribution is similar to the distribution found in [21], the mean length is 299.3 symbols.

3.3. Statistics of contour symbols

In this section we focus on parameters of the melodies itself. First, the distribution of symbols used in the melodies are a point of interest. Second, the statistical dependencies in the sequences of symbols are evaluated.

KIM and CHAI evaluated five step contours as they are used in MPEG-7[12, 14]. Their measurements show a uniform distribution. Also for our MDB the symbols are uniformly distributed, as shown in Figure 2.

More statistical parameters of the melodies taken as a sequence of symbols are obtained by measurement of the conditional probabilities. The sequence of contour values is a random process with discrete time and discrete values [26]. A simple type of such a process is the Markov chain of first order. The melody contour is a random process with $M = 5$ symbols s_m. Having the same number of states $u(k)$ at time k, that depend on the last state $u(k-1)$. The conditional probability for a change from state s_m to state s_n can be written as

$$P_{mn}(k) = P\left(u(k) = s_n | u(k-1) = s_m\right). \qquad (10)$$

There are M^2 possible state changes. The probabilities of all state changes can be summarised in a transition probability matrix $\mathbf{Q}(k)$ containing the elements

$$\mathbf{Q}(k) = [P_{mn}(k)]. \qquad (11)$$

Table 1 shows the transition probability matrix of the MIDI-MDB. The maximum value is $P(\text{»r«}|\text{»r«}) = 0.08$. This means a repetition »r« followed by a repetition »r« occurs most often. Next, $P(\text{»d«}|\text{»d«})$ and $P(\text{»u«}|\text{»u«})$ are high valued. In other words, a musical scale is often played upwards or downwards note by note. Interestingly, probabilities for $P(\text{»U«}|\text{»D«})$ and $P(\text{»D«}|\text{»U«})$ are

Table 1: Transition matrix **Q** with conditional probabilities of the MIDI melody database.

$P_{mn}(k) = P(u(k) = s_n \mid u(k-1) = s_m)$					
			s_n		
	D	d	r	u	U
D	0.019	0.037	0.027	0.029	0.059
d	0.025	0.080	0.043	0.053	0.037
s_m r	0.026	0.038	0.080	0.030	0.039
u	0.026	0.053	0.034	0.059	0.022
U	0.074	0.028	0.029	0.024	0.026

high. Since »D« and »U« are used for the interval minor third and more, jumping to the next chord note may be indicated by this probabilities.

3.4. Generation of random melody databases

For the generation of random melody databases the statistical properties of the MIDI melodies are used. To generate randomized melodies, the distribution function for melody lengths from Figure 1 and for symbol distribution in Figure 2 were used for a first approach. The contour symbols itself were generated without any statistical dependencies in this case. For a second approach, the database was built using the Markov-model employing the transition matrix in Table 1. The obtained MDBs are denoted as *Random-MDB* and *Markov-MDB* in the following sections.

4. EVALUATION

The result of a query will change if different MDBs with different properties are used. Obviously, the success of a query decreases if more titles are contained in the MDB, because more melodies may be similar to the query. Also, the genre of music may have a certain impact on the search results. Finally the distance metrics used in the QBH-system influence the result for a query.

DANNENBERG et al. evaluate databases with respect to the number of entries [23]. They use three different search algorithms working on some kinds of melody representations. In their results, they find slow decreasing success of queries with raising MDB sizes N_{DB}. To model this relationship they use the expression $R_{\mathrm{MRR}} = \frac{1}{\log N_{DB}}$ which represents the success of queries reasonably well. The value R_{MRR} indicates the *mean reciprocal rank* (MRR).

The typical usage of a QBH-system is to seek for the desired melody in a list of results. Therefore, in this paper the *recall* is used to judge the success of query.

Let $D = \{d_1,...,d_{N_{DB}}\}$ be the set of melodies in the database and D_q a subset of melodies found for query q. Furthermore, R_q is the a set of melodies from D_q, that are relevant for query q. The number of relevant documents is indicated by $|R_q|$. The recall is then defined as

$$V := \frac{|D_q \cap R_q|}{|R_q|}. \tag{12}$$

In our evaluation all melodies in the database are different, therefore V is either 1 or 0. To find out the recall for a distinct database size, e. g. $N_{DB} = |D|$ a set of T melodies is chosen from the database as a query. The number of queries found for query q is limited to $S = |D_q| = \{1,3,10\}$. The mean value of V_S indicates the performance of T queries. In this paper $T = 20$ is chosen.

4.1. Melody databases

First, the search performance of the MDB created from MIDI data is evaluated. Second, the same evaluation is carried out with MDBs containing randomized melodies.

4.1.1. Melody contours from MIDI

For evaluation three sets of $T = 20$ melodies of the MIDI-MDB are selected and the mean recall values V_S are taken for different database sizes. The length of the queries is limited to 8 symbols. This is a critical value as indicated by formerly performed evaluations [11]. For shorter queries no useful results are retrieved, for longer queries the differences between the different distance measures become less detailed. The MDB size is limited to $N_{\mathrm{DB}} = 20, 50, 100, 200, 300$ and 400 to observe the different performances depending on the database size. A size of $N = 1000$ is used for the randomized MDBs. This size is a first attempt to see if the modellation of the database content results in a resonable performance similiar to the MIDI-MDB. Much larger sizes are required for real world szeanrios, as pointed out in Section 3.1. The distance metrics are UK, SF, CM and LCE, LCT and LAL using optimum normalisation (see Section 2).

In Figure 3 the recall V_1 for all distance metrics are shown, marked by an asterisk. SF, CM, LAL and LCT perform same and better than UK, and LCE yields unusable results. Results are much better for V_3 as shown in Figure 4(a).

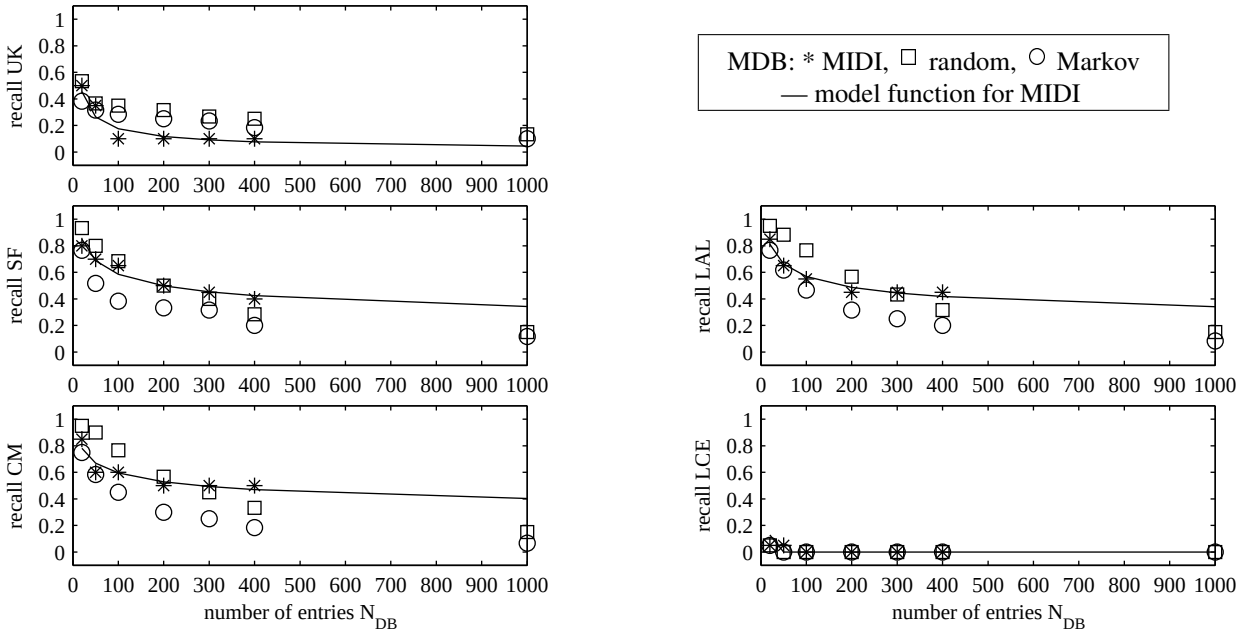


Figure 3: Recall V_1. Please note that LAL and LCT yield exactly the same results. Therefore only results for LAL are depicted.

Again SF, CM, LAL and LCT perform best followed by UK. Due to poor performance LCE is left out here. Lastly, V_{10} is depicted in Figure 4(b). The overall results are better than for V_3, LCE does not perform reasonable again.

To model the behaviour of the recall values R_{MIDI} of the MIDI-MDB, the non linear model function

$$R_{\mathrm{model}}(n) = \begin{cases} 1 & \text{if} \quad R_{\mathrm{MIDI}} = 1 \\ c_1\, n^{c_2} & \text{if} \quad R_{\mathrm{MIDI}} < 1 \end{cases} \qquad (13)$$

with some constants c_1 and c_2 is used. The constants are obtained taking all recall values less than 1 and using a minimum mean square error criterion. Values of the model function are indicated by a solid line in Figure 3, 4(a) and 4(b).

4.1.2. Contours from random melodies

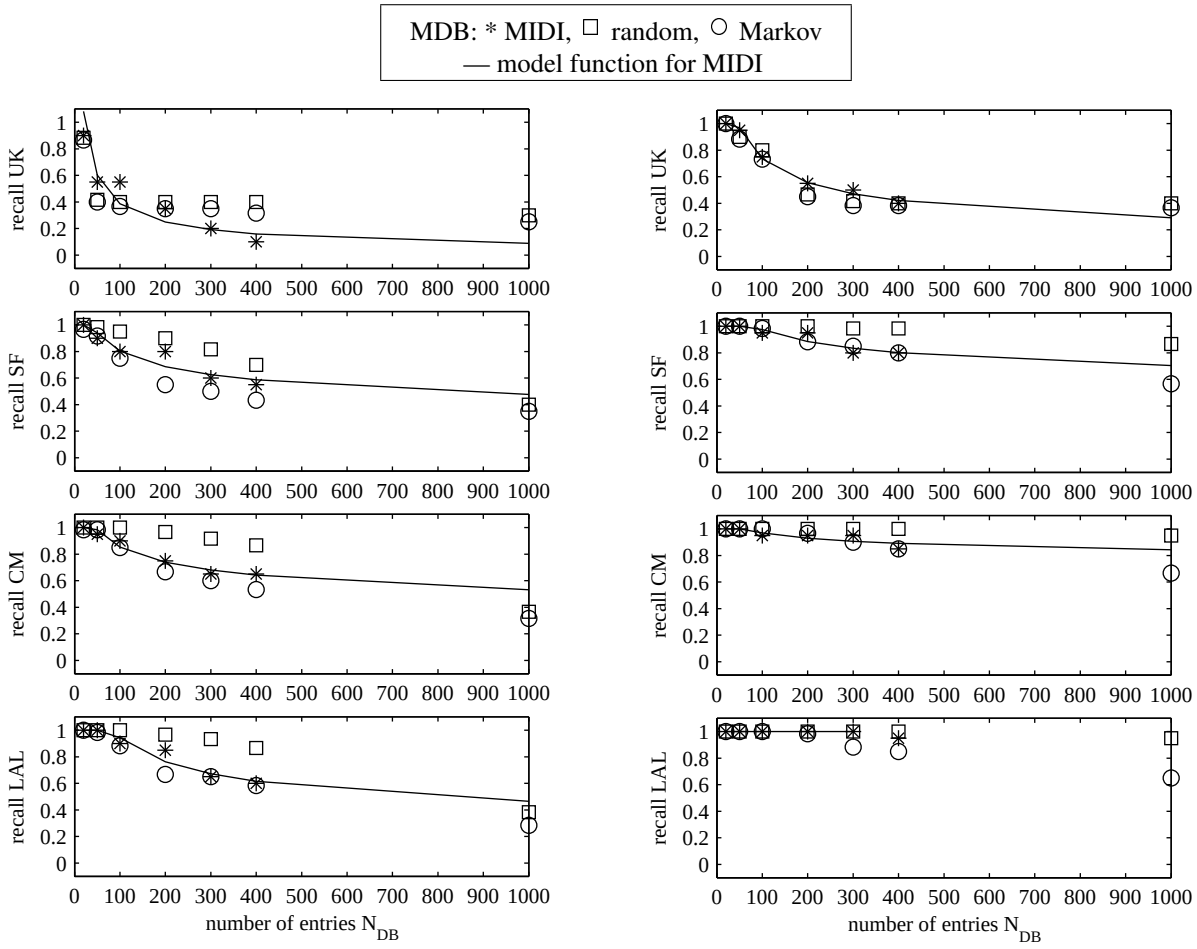
The Random-MDB and Markov-MDB were evaluated the same way as the MIDI-MDB. Again three sets of 20 titles were chosen from the respective databases and limited to 8 symbols.

For V_1 (see Figure 3) the Random-MDB fits better to the values of the MIDI-MDB for SF and CM, whereas the recall values using the Markov-MDB fit better for UK. For all other distance metrics, both databases fit to the values of the MIDI-MDB. For V_3 and V_{10} the Markov-MDB yields results that are a better approximation for the results of the MIDI-MDB than using the Random-MDB, see Figure 4(a) and 4(b).

The recall values V_1 of the ramdomized databases at $N_{\mathrm{DB}} = 1000$ fit well for UK and LCE with the results obtained by the model function (13), but SF, CM, LAL and LCT yield smaller recall values. For V_3, except for UK all recall values for $N_{\mathrm{DB}} = 1000$ are smaller than obtained by the model function. For V_{10} the recall values of the Markov-MDB are higher than the model function (13). For LAL no extrapolation is possible since only one recall value of the MIDI recall values is less than 1.

4.2. Discussion

Especially for V_3 and V_{10} the Markov-MDB containing random melodies appears to be a sufficient replacement for a MIDI-MDB with authentic melodies. The Markov-MDB is generated using just three statistical parameters, that are distribution of the melody lengths, distribution of contour symbols and the state transition matrix to model



(a) Recall V_3 (b) Recall V_{10}

Figure 4: Recall values for V_3 and V_{10}. LCE is omitted due to poor performance. Results of LAL and LCT are identical.

the Markov process. Even the Random-MDB using only the distributions of length and symbols follows the trend of results obtained by the MIDI-MDB pretty well.

A non linear model function is used to extrapolate the behaviour of the MIDI-MDB, see equation (13). However, the results of the randomized MDBs do not match well with the model for larger values of N_{DB}.

5. CONCLUSIONS

The performance of a QBH-system was evaluated in this paper by measuring the recall. All measurements were performed with the original MIDI-MDB using 400 melody contours and two newly generated randomized MDBs containing 1000 entries. For generation of the Random-MDB only the distribution of the melody lengths and the distribution of the contour symbols were used, the Markov-MDB employed also a state transition matrix derived from the existing melody contours to model a Markov process of order one.

Recall values were measured using different query sets. For query generation 20 melodies were taken from the respective MDBs and truncated to get short queries that are typical for use with QBH-systems. The recall values obtained were similar for MIDI and random filled MDBs, especially in case of the Markov-MDB. To check the results of the randomized MDBs, the recall values of the MIDI-MDB were extrapolated using a non linear modelling function. It turned out that recall values of the

model and randomized MDBs do not match in all cases, therefore further work is necessary on this topic.

A notable result of the experiments performed is that the performance of a QBH-system is independent in some sense of the database's content. On the other hand, the Markov model improves the convergence of results for random melodies and authentic melodies. Further investigations could include the use of higher order models. Generally speaking to model the data in the MDB is a promising utility for evaluation of distance metrics used in the QBH-system. The scaleability in size enables the evaluation of real world scenarios.

6. REFERENCES

[1] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith, "Query by Humming: Musical Information Retrieval in an Audio Database," in *ACM Multimedia*, 1995, pp. 231–236.

[2] G. Haus and E. Pollastri, "An Audio Front End for Query-by-Humming Systems," in *2nd Annual International Symposium on Music Information Retrieval*. Bloomington, Indiana, USA: ISMIR, 2001.

[3] L. P. Clarisse, J. P. Martens, M. Lesaffre, B. D. Baets, H. D. Meyer, and M. Leman, "An Auditory Model Based Transcriber of Singing Sequences," in *Proceedings of the ISMIR*, 2002, pp. 116–123.

[4] L. Prechelt and R. Typke, "An interface for melody input," *ACM Transactions on Computer-Human Interaction*, vol. 8, no. 2, pp. 133–149, 2001.

[5] A. L. Uitdenbogerd, "Music Information Retrieval Technology," Dissertation, Royal Melbourne Institute of Technology, May 2002.

[6] J.-M. Batke, G. Eisenberg, P. Weishaupt, and T. Sikora, "Evaluation of Distance Measures for MPEG-7 Melody Contours," in *International Workshop on Multimedia Signal Processing*, Siena, Italy, Oct. 2004.

[7] R. B. Dannenberg and N. Hu, "Understanding Search Performance in Query-By-Humming Systems," in *Proc. of the ISMIR*, 2004.

[8] S. Downie, "Evaluation of a Simple and Effective Music Information Retrieval Method," in *SIGIR*. ACM, 2000.

[9] R. J. McNab and L. A. Smith, "Evaluation of a melody transcription system," in *Proceedings of the ICASSP*, 2000, pp. 819–822.

[10] T. D. Mulder, J.-P. Martens, M. Lesaffre, M. Leman, B. D. Baets, and H. D. Meyer, "An Auditory Model Based Transcriber of Vocal Queries," in *Proc. of the ISMIR*, 2003.

[11] J.-M. Batke, "Untersuchung von Melodiesuchsystemen sowie von Verfahren zu ihrer Funktionsprüfung," Dissertation, Technische Universität Berlin, 2006, to be published.

[12] Y. E. Kim, W. Chai, R. Garcia, and B. Vercoe, "Analysis of a Contour-Based Representation for Melody," in *Proc. International Symposium on Music Information Retrieval*, Oct. 2000.

[13] E. Brochu and N. de Freitas, "Name That Song!: A Probabilistic Approach to Querying on Music and Text," in *Neural Information Processing Systems*, 2002, pp. 1505–1512.

[14] W. Chai and B. Vercoe, "Melody Retrieval On The Web," in *Proceedings of ACM/SPIE Conference on Multimedia Computing and Networking*, Jan. 2002.

[15] G. Eisenberg, J.-M. Batke, and T. Sikora, "Efficiently Computable Similarity Measures for Query by Tapping Systems," in *Proc. of the 7th Int. Conference on Digital Audio Effects*. Naples, Italy: DAFx, Oct. 2004, pp. 189–193.

[16] S. Pauws, "CubyHum: A Fully Operational Query by Humming System," in *Proc. of the 3rd ISMIR*, Oct. 2002.

[17] L. Lu, H. You, and H.-J. Zhang, "A new approach To Query By humming In Music Retrieval," in *IEEE International Conference on Multimedia and Expo*, 2001.

[18] Y. Feng, Y. Zhuang, and Y. Pan, "A Hierarchical Approach: Query Large Music Database by Acoustic Input," in *SIGIR'02*, 2002, pp. 441–442.

[19] Y. Zhu and M. Kankanhalli, "Music Scale Modelling for Melody Matching," in *Proc. MM*, 2003, pp. 359–362.

[20] R. J. McNab, L. A. Smith, D. Bainbridge, and I. H. Witten, "The New Zealand Digital Library MELody inDEX," *D-Lib Magazine*, May 1997.

[21] N. Kosugi, Y. Nishihara, T. Sakata, M. Yamamuro, and K. Kushima, "A Practical Query-By-Humming System for a Large Music Database," in *Proceedings of the Eighth ACM International Conference on Multimedia*, 2000, pp. 333–342.

[22] T. Brondsted, S. Augustensen, B. Fisker, C. Hansen, J. Klitgaard, L. W. Nielsen, and T. Rasmussen, "A System for Recognition of Hummed Tunes," in *Proc. of the COST G-6 Conf. on DAFX*, Dec. 2001.

[23] R. B. Dannenberg, W. P. Birmingham, G. Tzanetakis, C. Meek, N. Hu, and B. Pardo, "The MUSART Testbed for Query-by-Humming Evaluation," *Computer Music Journal*, 2003.

[24] "Apple iTunes Overview," www.apple.com/itunes/overview, accessed on 2006-03-11.

[25] "mp3.com," www.mp3.com/feature/aboutus, accessed on 2006-03-11.

[26] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. New York: McGraw-Hill, 1965.