Hum-a-song: A Subsequence Matching with Gaps-Range-Tolerances Query-By-Humming System

Alexios Kotsifakos [†1], Panagiotis Papapetrou [*2], Jaakko Hollmén [*3], Dimitrios Gunopulos [#4], Vassilis Athitsos [†5], and George Kollios [‡6]

[†]*Department of Computer Science and Engineering, University of Texas at Arlington, USA*
[1]alexios.kotsifakos@mavs.uta.edu, [5]athitsos@uta.edu

[*]*Department of Information and Computer Science, Aalto University, Finland*
[2]panagiotis.papapetrou@aalto.fi [3]jaakko.hollmen@aalto.fi

[#]*Department of Informatics and Telecommunications, University of Athens, Greece*
[4]dg@di.uoa.gr

[‡]*Department of Computer Science, Boston University, USA*
[6]gkollios@cs.bu.edu

ABSTRACT

We present "Hum-a-song", a system built for music retrieval, and particularly for the Query-By-Humming (QBH) application. According to QBH, the user is able to hum a part of a song that she recalls and would like to learn what this song is, or find other songs similar to it in a large music repository. We present a simple yet efficient approach that maps the problem to time series subsequence matching. The query and the database songs are represented as 2-dimensional time series conveying information about the pitch and the duration of the notes. Then, since the query is a short sequence and we want to find its best match that may start and end anywhere in the database, subsequence matching methods are suitable for this task. In this demo, we present a system that employs and exposes to the user a variety of state-of-the-art dynamic programming methods, including a newly proposed efficient method named SMBGT that is robust to noise and considers all intrinsic problems in QBH; it allows variable tolerance levels when matching elements, where tolerances are defined as functions of the compared sequences, gaps in both the query and target sequences, and bounds the matching length and (optionally) the minimum number of matched elements. Our system is intended to become open source, which is to the best of our knowledge the first non-commercial effort trying to solve QBH with a variety of methods, and that also approaches the problem from the time series perspective.

1. INTRODUCTION

During the last decades the problem of *subsequence matching* is of particular interest and has been studied by the database and data mining communities. Subsequence matching is the problem of finding the subsequence of a long sequences' database with arbitrary start and end best matching a much smaller query sequence.

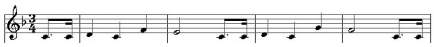
Proceedings of the VLDB Endowment, Vol. 5, No. 12

Although several distance or similarity based measures have been proposed in the literature for solving the subsequence matching problem in application domains such as time series, categorical sequences and multimedia data (Dynamic Programming (DP) methods are quite common [1]), in the music retrieval domain do not perform well or, much worse, cannot be applied or achieve very low retrieval accuracy. This is because music retrieval has many intrinsic characteristics that should be considered by a similarity or distance based measure. More specifically, the problem we are interested in is: assume that you would like to find in a large music repository a song you hear (or songs similar to it), because you cannot recall its name. One obvious way to do so, is to hum the part of the melody that you remember and then perform a similarity search over the database by using a time series representation. This way, the problem of *Query-By-Humming* (QBH) becomes a subsequence matching problem in the time series domain. We call our demo system "Hum-a-song", which given a hummed query song by a user searches a large music database to identify the top-K most similar songs to the sung melody, by allowing the user to select among a variety of subsequence matching methods.

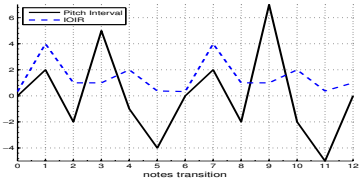
Next, we provide some basic and important music terms, which will help in understanding how time series subsequence matching can be applied to QBH. The pattern of allowed intervals that a song's sequence of notes should comply with is called *key*, and the speed of every music piece is characterized by the *tempo*. In addition, each *note* in a music piece is defined by its *pitch* and *duration*, and the difference in frequency of two adjacent notes is called *pitch interval*. Moreover, *semitone* is the smallest pitch interval, and an *octave* is defined by 12 consecutive semitones. *Transposition*, which is significant when representing music pieces as time series, is defined as shifting a melody from one key to another.

As shown in [9], both pitch and duration should be used for representing musical pieces, as there may be the case where many songs have quite similar pitch values but their notes' durations differ. To avoid the possibility of erroneously matching songs, duration information is also taken into account in our system; melodies are represented by 2-dimensional time series of notes of arbitrary length, where the first dimension corresponds to pitch and the second one to duration (Figure 1).

Our first contribution is the "Hum-a-song" QBH system, that allows the user to record a melody that she recalls and get the top-K



(a) Part of the music score.



(b) Representation using pitch intervals and IOIR.

Figure 1: Example of the music score and its 2-dimensional time series representation. IOIR is the duration ratio of two consecutive notes.

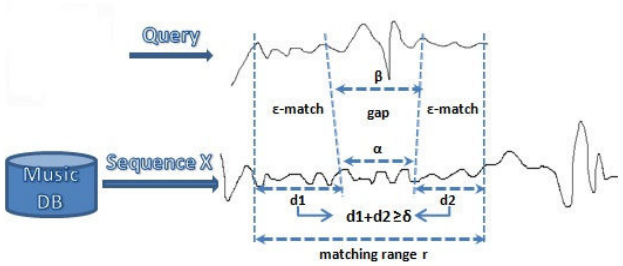


Figure 2: SMBGT: error-tolerant matching is denoted as ϵ-match.

most similar to this melody songs. The system gives the opportunity to the user to select among various similarity or distance based methods to assess the similarity between the query and the database songs. Secondly, since QBH is a very noisy domain, we are interested in performing robust and efficient subsequence matching. Consequently, apart from the fact that the user is given the opportunity to test and compare several methods, a newly proposed method is also implemented and included in our system, named *SMBGT* [5]. SMBGT is an *error-tolerant* subsequence matching method that finds the subsequence of the target sequence X that best matches the query song Q (where $|Q| << |X|$) and is capable of handling a variety of errors when the user hums a song, such as instant or temporary key or tempo loss. This is achieved by the features that it encompasses. To begin with, it allows *variable tolerance levels* in the matching (where tolerances are defined as functions of the query and target 2-dimensional elements). Also, it allows skipping a bounded number of consecutive elements on both the query and target time series, as defined by β and α, respectively. What is more, the maximum match length in X is constrained by r, and optionally imposes an additional constraint δ to the number of elements matching in the query and target sequences, based on the singing skills of the user. An example of SMBGT is shown in Figure 2. Furthermore, our system is not a commercial product (such as Shazam[1], Midomi[2] and Soundhound[3]) and is intended to become open source, so that it can be extended to include, for example, more similarity/distance measures for comparison and benchmarking purposes. Last but not least, since we approach the QBH problem by mapping it to the time series domain, the system can be easily extended for other time series application domains in which subsequence matching would be of interest, such as finding specific patterns in sensor, financial, weather data among others.

[1]http://www.shazam.com/music/web/home.html
[2]http://www.midomi.com/
[3]http://www.soundhound.com/

2. PROBLEM SETTING

2.1 Encoding Pitch and Duration

Regarding pitch, there are two well-known ways of expressing it, the *absolute pitch*, and the *pitch interval*. In absolute pitch the frequency of the note is used, and this value in MIDI is an integer in $[1, 127]$ (0 corresponds to *pause*), while in pitch interval the first dimension is the difference in frequencies between two subsequent notes. Referring to the second dimension, duration, it can be encoded in three different ways [6], *Inter-Onset-Interval* (IOI) which is the difference in time onsets/clicks of two consecutive notes, *IOI Ratio* (IOIR), that is the ratio of the IOIs of two subsequent notes (the IOIR of the last note is 1), and *Log IOI Ratio* (LogIOIR), defined as the logarithm of the IOIR.

2.2 Variable Tolerances

Let $X = \{x_1, \ldots, x_n\}$ be a time series of length $|X|$ that corresponds to a melody, where $x_j = \langle x_j^p, x_j^r \rangle \in X$ is a pair such that x_j^p maps to pitch and x_j^r to duration information. A set of N such time series constitutes a music database $DB = \{X_1, \ldots, X_N\}$. A *subsequence* of X, $X[ts : te] = \{x_{ts}, \ldots, x_{te}\}$, is comprised by a set of 2D points, the indices of which appear in the same order as in X and are not necessarily continuous. Also, let query $Q = \{q_1, \ldots, q_m\}$ be a 2D time series. Since in QBH it is quite usual for users to make instant humming errors [7], error-tolerant matches should be allowed. We say that $q_i \in Q$ and $x_j \in X$ *match with variable ϵ-tolerance*, $q_i \approx_\epsilon^f x_j$, if, we either use absolute or relative tolerance, and for $\epsilon^f = \{\epsilon_p^f, \epsilon_r^f\}$, it holds that $\epsilon_p^f(i) = f_p(q_i^p)$ and $\epsilon_r^f(i,j) = f_r(q_i^r, x_j^r)$, where f_p is a function of q_i^p and f_r a set of constraints on q_i^r, x_j^r. It should be mentioned that $\epsilon_p^f, \epsilon_r^f$ can be constant as well, depending on the application domain or the user of our system.

For our QBH system, ϵ_p^f can be defined as

$$\epsilon_p^f(i) = \lceil q_i^p * t \rceil, t \geq 0 \tag{1}$$

and can be used with both absolute and relative tolerances [5]. Referring to $\epsilon_r^f(i,j)$, for IOIR representation the scheme used is

$$\epsilon_r^f(i,j) = \{x_j^r \leq 2 * q_i^r,\ x_j^r - q_i^r \geq -0.5\} \tag{2}$$

and for LogIOIR

$$\epsilon_r^f(i,j) = \begin{cases} \{0 \leq log_2(x_j^r/q_i^r) \leq 1\}, log_2 x_j^r \geq 0 \\ \{|log_2(x_j^r/q_i^r)| \leq 1\}, log_2 x_j^r < 0 \end{cases} \tag{3}$$

2.3 Problem Formulation

The problem formulation follows [5]. Given a database DB, a query Q, and positive integers δ and r, it identifies the set $\mathcal{S} = \{X_i[ts : te] | X_i \in DB\}$ of the top-K subsequences for which $|SMBGT(Q, X_i[ts : te])| \geq \delta$. $SMBGT(Q,X)$ is the longest common bounded gapped subsequence of Q and X. In other words, it is the pair $\{Q[ts_1{:}te_1], X[ts_2{:}te_2]\}$, where the subsequences are of the same length that is the maximum possible one, each element of $Q[ts_1{:}te_1]$ matches with variable ϵ-tolerance with one element of $X[ts_2{:}te_2]$ in order, consecutive indices of the elements of $Q[ts_1{:}te_1]$ and $X[ts_2{:}te_2]$ should not differ by more than α and β, respectively, and it also holds that $te_2 - ts_2 \leq r$.

3. HUM-A-SONG - DEMO

3.1 Recording Queries

The user of the system is first asked to record the part of the song she recalls and is interested in finding in the database. For this

Table 1: Code numbers for 2D representations

Code number	Representation
1	⟨mod12, IOIR⟩
2	⟨mod12, LogIOIR⟩
3	⟨mod12, LogIOIR in $[-2, 2]$⟩)
4	⟨mod12, LogIOIR quantized to closest integer⟩
5	⟨pitch interval, IOIR⟩
6	⟨pitch interval, LogIOIR⟩
7	⟨pitch interval, LogIOIR in $[-2, 2]$⟩)
8	⟨pitch interval, IOIR quantized to closest integer⟩

purpose, the *Akoff music composer-version 2.0* [4], a tool commonly used for evaluating QBH systems is used [11]. The user is asked to hum the part of song close to a microphone, and to avoid singing with lyrics. Then, the melody sang is converted to MIDI through Akoff. Finally, in order to get a 2D time series from the melody, the pitch at every time click is extracted, and tuples ⟨pitch, click⟩ are converted to representation 5 of Table 1. It is noted that the user can also perform an experiment on a pre-recorded hummed query.

3.2 Representation

After recording a query song, the user is able to select among a variety of 2D time series representations. For the purposes of our demo, and since we are interested in note transitions, so as not to check all possible transpositions of a melody nor to scale in time when comparing time series, we consider the following encoding schemes: ⟨pitch interval, IOIR⟩ and ⟨pitch interval, LogIOIR⟩. In Figure 1(b) we can see an example of the ⟨pitch interval, IOIR⟩ representation of the part of "Happy Birthday" song shown in Figure 1(a). Except for these schemes, the user is allowed to choose among some other options. First, by applying modulo 12 to the pitch intervals they are transformed/quantized to $[-11, 11]$ [9]. Mapping the pitch intervals into two octaves is a very reasonable quantization, as the human singing range rarely goes beyond this range. Second, the LogIOIR encoding of time can be quantized to the closest integer or the closest value in $[-2, 2]$ [6]. The transposition and time invariant representations are shown in Table 1.

3.3 Method Selection

Since we are interested in evaluating a QBH system, apart from the SMBGT and SMGT methods [5], the user is allowed to choose among a variety of methods that can be applied to music retrieval. These methods, performing DP-based subsequence matching, are SPRING [8], a version of Edit distance suitable for music retrieval [10], and two DTW-based [3, 4] (denoted DTW_s and DTW_c). We note that Edit has been modified to account for LogIOIR and quantizations (Table 1), and SPRING allows for varying r [5].

3.4 Tuning Parameters

Having selected the query, the representation, and the method, the user has to provide the appropriate parameters for the requested method. For SMBGT, a novel method for subsequence matching [5], the number of consecutive gaps allowed in *both* X and Q has to be given, which are identified by the α and β positive integers, respectively. The intuition for the existence of these two constraints is that there may be serious humming errors if there is temporary key/temp loss or significant instant note loss that cannot be treated by the accepted tolerance. Also, parameter r is used to eliminate large matches, e.g., $r = 1.2 * |Q|$, δ to bound the minimum number of matching elements, e.g., $0.5 * |Q|$, and variable tolerance can be specified combined with either absolute or relative tolerance scheme. If the user selects SMGT then the input parameters are the same, except that no constraint is imposed on the lengths of the allowed gaps. With regard to SPRING, since it has been modified not to account for infinite matching lengths, the user has to provide r, as happens with SMBGT and SMGT. Finally, DTW_s and DTW_c require the initialization parameter c, while the version of Edit distance used [5] does not require any input parameter.

3.5 Evaluation

Below, we present the final steps for getting the final result set for the hummed query. However, for completeness and demo purposes, our system has some pre-inserted synthetic and hummed query sets (that the user may use to compare accuracies between different methods), instead of humming a query.

3.5.1 Database

The music database on which we test our system consists of 5640 MIDI[5] files freely available on the web. It covers several music genres such as classical, blues, jazz, rock, rock 'n' roll, pop, and also themes from tv and movies series. To obtain the 2D time series representation, for each of the (at most) 16 channels that every song is comprised, the highest pitch at every time click was extracted-*all-channels extraction* [9]. Then, each channel's tuples ⟨pitch, click⟩ were converted to representation 5 of Table 1. This step generated 40891 2D time series loaded in memory at runtime.

3.5.2 Hummed Query

The user also has to insert the number of top-K songs she wishes to be returned in the result set for the melody sung. The top-K results appear in the final screen of our system, along with the 2D time series plot of the given melody with the top candidate and the time for returning the results. What is more, the user can select and listen to a song appearing in the list of results, listen to the query, see more details regarding the selected song, i.e., similarity/distance score, channel giving that score, start and end points of the matching, and also the 2D plot of the song's channel giving the best score. The performance of the system depends on the hummed query, the method, and the parameters given. For details about the methods' evaluation please refer to [5]. The whole procedure can be certainly repeated by recording another melody (*Main Menu* button). An example of setting an experiment for our system along with the experimental results is shown in Figure 3.

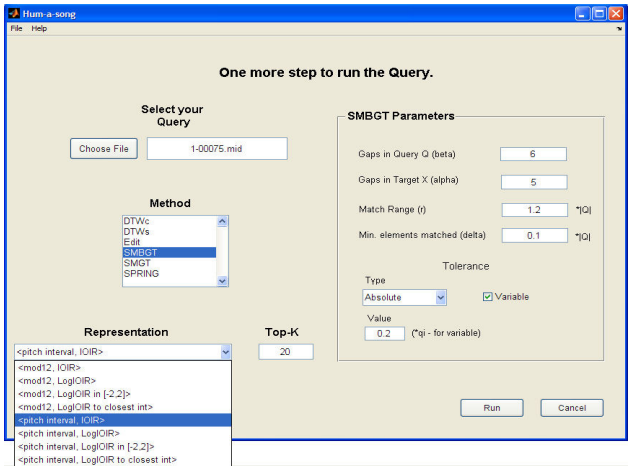
3.5.3 Testbeds

Another possibility offered by the proposed system is the testbeds, comprising five synthetic query sets and one hummed.

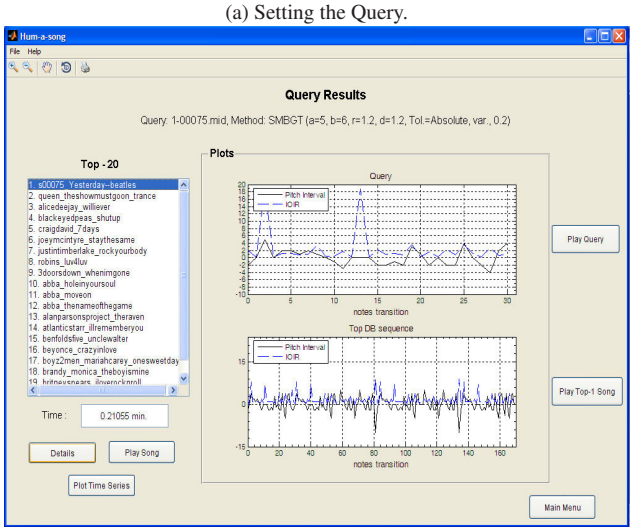
Each of the five synthetic query sets includes 100 queries of lengths ranging in $[13, 137]$, and are denoted as $Q_{.10}$, $Q_{.20}$, $Q_{.30}$, $Q_{.40}$, and $Q_{.50}$. Sets $Q_{.10} - Q_{.50}$ were created by modifying randomly the 10, 20, 30, 40, and 50% of the 2D points of exact segments of the database. The pitch interval of the selected points was changed by $\pm k \in [3, 8]$, which is a reasonable value, so that the error inserted by a user when humming and also the noise that may be added by any audio processing tool is well simulated. Regarding IOIR, for each of the picked points, it was changed by $\pm k \in [2, 4]$, so that different duration ratios could exist and in addition not be biased against SMBGT and SMGT as shown by Equation 2. Moreover, at most 3 adjacent elements in all query sets were modified, since by having more consecutive errors would increase false positives. The hummed query set consists of 100 hummed melodies of lengths 14 to 76 covering various music genres, such as classical ("Für Elise"), blues ("Hideaway"), jazz ("Strangers in the Night"),

[4] http://www.akoff.com/music-composer.html

[5] Musical Instrument Digital Interface



(a) Setting the Query.



(b) Query results.

Figure 3: "Hum-a-song": Screenshots of setting the experiment and getting the results for SMBGT, where $|Q| = 31$, the database has 4000 sequences, and the target "Yesterday" song is the Top-1. For the target sequence, $Score = 20$, $Startpoint = 28$, $Endpoint = 56$.

rock 'n' roll ("Rock Around the Clock"), rock ("Fly Away"), country ("Hey Good Lookin"), and romantic songs ("What a Wonderful World"). These queries were hummed using a microphone and then converted to MIDI using Akoff.

When choosing a testbed, in the final screen, apart from the time required for the results to be returned, the user is presented with three evaluation measures for the method and representation selected: *recall*, which is the % of queries that have their correct answer in the top-K results, *mean reciprocal rank (MRR)* [2], which is the mean inverse rank of all queries in their top-K results, and the *average rank (AR)* of all queries in the selected query set. It has to be mentioned that all measures are of particular importance for evaluating a QBH method, since recall indicates if the method identifies the correct answer in the top-K results, and MRR along with average rank show if recall can be improved by decreasing K. The representation, number of top-K results used, and the parameters of the selected method giving the best recall are also shown.

Our demo system is implemented using Matlab and currently runs in Windows Operating System. It can be downloaded from http://vlm1.uta.edu/∼akotsif/hum-a-song/.

4. CONCLUSIONS

Motivated by the QBH application, we developed a demo system named "Hum-a-song" that allows the user to hum a part of a song that she recalls and find the most similar to this songs. This is done by transforming this problem to the time series domain, where the query and the songs of the music database are mapped to 2D time series and then applying a subsequence matching method to get the subsequence of the database that best matches the query. Several such methods have been implemented and given as an option to the user, including the robust and efficient SMBGT method. Finally, our system can be easily extended to other time series application domains, basically due to the fact that the queries and the target sequences are represented as time series and the methods employed perform on top of such representation.

Acknowledgments

This work has been partially supported by the Academy of Finland Algorithmic Data Analysis (ALGODAN) Centre of Excellence grant 118653, the MODAP and DISFER projects, the NSF grant IIS-0812309, and by NSF grants IIS-0812601, IIS-1055062, CNS-0923494, and CNS-1035913.

5. REFERENCES

[1] R. Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.

[2] R. Dannenberg, W. Birmingham, G. Tzanetakis, C. Meek, N. Hu, and B. Pardo. The MUSART testbed for query-by-humming evaluation. *Computer Music Journal*, 28(2):34–48, 2004.

[3] N. Hu, R. Dannenberg, and A. Lewis. A probabilistic model of melodic similarity. In *ICMC*, pages 509–515, 2002.

[4] J. Jang and M. Gao. A query-by-singing system based on dynamic programming. In *International Workshop on Intelligent Systems Resolutions*, pages 85–89, 2000.

[5] A. Kotsifakos, P. Papapetrou, J. Hollmén, and D. Gunopulos. A subsequence matching with gaps-range-tolerances framework: a query-by-humming application. *Proceedings of Very Large Data Bases*, 4(11):761–771, 2011.

[6] B. Pardo and W. Birmingham. Encoding timing information for musical query matching. In *ISMIR*, pages 267–268, 2002.

[7] B. Pardo, J. Shifrin, and W. Birmingham. Name that tune: A pilot study in finding a melody from a sung query. *Journal of the American Society for Information Science and Technology*, 55(4):283–300, 2004.

[8] Y. Sakurai, C. Faloutsos, and M. Yamamuro. Stream monitoring under the time warping distance. In *ICDE*, pages 1046–1055, 2007.

[9] A. Uitdenbogerd and J. Zobel. Melodic matching techniques for large music databases. In *ACM Multimedia (Part 1)*, page 66, 1999.

[10] E. Unal, E. Chew, P. Georgiou, and S. Narayanan. Challenging uncertainty in query by humming systems: a fingerprinting approach. *Transactions on Audio Speech and Language Processing*, 16(2):359–371, 2008.

[11] Y. Zhu and D. Shasha. Warping indexes with envelope transforms for query by humming. In *SIGMOD*, pages 181–192, 2003.