Benchmarking Dynamic Time Warping for Music Retrieval

Jefrey Lijffijt[1,2], Panagiotis Papapetrou[1,2], Jaakko Hollmén[1,2], and Vassilis Athitsos[3]

[1] *Department of Information and Computer Science, Aalto University School of Science and Technology, Finland*
[2] *Helsinki Institute for Information Technology, Finland*
[3] *Computer Science and Engineering Department, University of Texas at Arlington, USA*

ABSTRACT

We study the performance of three dynamic programming methods on music retrieval. The methods are designed for time series matching but can be directly applied to retrieval of music. Dynamic Time Warping (DTW) identifies an optimal alignment between two time series, and computes the matching cost corresponding to that alignment. Significant speed-ups can be achieved by constrained Dynamic Time Warping (cDTW), which narrows down the set of positions in one time series that can be matched with specific positions in the other time series. Both methods are designed for full sequence matching but can also be applied for subsequence matching, by using a sliding window over each database sequence to compute a matching score for each database subsequence. In addition, SPRING is a dynamic programming approach designed for subsequence matching, where the query is matched with a database subsequence without requiring the match length to be equal to the query length. SPRING has a lower computational cost than DTW and cDTW. Our database consists of a set of MIDI files taken from the web. Each MIDI file has been converted to a 2-dimensional time series, taking into account both note pitches and durations. We have used synthetic queries of fixed size and different noise levels. Surprisingly, when looking for the top-K best matches, all three approaches show similar behavior in terms of retrieval accuracy for small values of K. This suggests that for the specific application area, a computationally cheaper method, such as SPRING, is sufficient to retrieve the best top-K matches.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

time series, query-by-humming, dynamic time warping.

1. INTRODUCTION

Time series data occur in a wide range of real-world applications and are used to represent a variety of data domains, such as scientific measurements, financial data, music, human activity, etc. Thus, in multiple domains, large databases of sequences are used as knowledge repositories. At the same time, retrieving information of interest in such repositories becomes a challenging task, due to the large amounts of data that need to be searched.

Subsequence matching is the problem of identifying, given a query sequence and a database of sequences, the database *subsequence* that best matches the query sequence. Achieving efficient subsequence matching is an important problem in domains where the database sequences are much longer than the queries, and where the best subsequence match for a query can start and end at any position of any database sequence.

One such domain is music. We assume that the knowledge repository is a database of music pieces. Every music piece consists of notes, with each note being described by a *pitch* (which corresponds to the sound frequency of the note) and a *duration*. An example of a music piece is shown in Figure 1, where we can see the music score of the song *Happy Birthday to You*. Pitch and duration are two distinctive characteristics for a music piece and determine its melody; hence, in order to provide a sufficient music representation, we consider a music piece to be a sequence of pitches with their corresponding durations.

Since the melody of two music pieces can be the same but in different tempo, or in higher frequency, both pitch and duration should be properly normalized. Thus, instead of the absolute pitch of each note, we use the *Relative Pitch* (RP), which is the pitch difference of two adjacent notes. The note duration is represented using the *Duration Ratio* (DR), which is the ratio of the durations of two adjacent notes. The DR of the last note is 1. This type of normalization has also been used in several music retrieval systems [29, 10, 38, 26].

Using the above representation, each music piece is defined as a 2-dimensional time series $X = \{(p_i, d_i)), i = 1, \ldots, |X|\}$, where p_i the relative pitch of each pair of consecutive notes and d_i is their respective duration ratio. A query sequence corresponds to a music piece that is either hummed or played by some music instrument. At runtime, the query is also transformed to a 2-dimensional time series using the same representation. In this setting, existing time series subsequence matching methods can be applied directly to retrieve the best match of the query in a music database.

Typically, similarity between sequences is measured us-

Figure 1: A music sheet of the music score for the song *happy birthday to you* taken from `http://www.8notes.com`**.**

ing algorithms based on dynamic programming (DP) [4]. In particular, dynamic time warping (DTW) [16] is widely used for time series data. Constrained Dynamic Time Warping (cDTW) is a modification of DTW that places constraints on the possible alignment between two sequences [32]. These DP-based algorithms can be used both for full sequence and for subsequence matching, and identify the globally optimal subsequence match for a given query [23, 25, 33]. While this complexity is definitely attractive compared to exhaustively matching the query with every possible database subsequence, in practice subsequence matching is still a computationally expensive operation in many real-world applications, especially in the presence of large database sizes.

In this paper, we study the performance of three existing time series subsequence matching methods on music retrieval. We assume that each query can be of arbitrary length, i.e., it can be a part of the melody of a music piece we are looking for in our database and not necessarily the whole piece. This is a more realistic setting for the field of music retrieval as people usually tend to remember parts of the melody of a music song and not the whole melody. Under this assumption, we study three major time series matching algorithms: Dynamic Time Warping (DTW) [16], constrained Dynamic Time Warping (cDTW) [32], and SPRING [33]. Since the first two are designed for full sequence matching, we use a sliding window over the whole database, equal to the query length, to compute the matching score of each possible subsequence.

2. RELATED WORK

A large body of literature addresses the problem of sequence matching. Several methods assume that sequence similarity is measured using the Euclidean distance [11, 7, 21, 22] or variants [1, 30, 41]. However, such methods cannot handle even the smallest misalignment caused by time warps, insertions, or deletions. Robustness to misalignments is achieved using distance measures based on dynamic programming, such as dynamic time warping (DTW) [16] and edit distance based approaches [19, 39, 8, 17, 9, 24]. In the remaining discussion we restrict our attention to the dynamic time warping distance measure, which is the most popular measure for time series.

Sequence matching methods can be divided into two categories: (1) methods for full sequence matching, where the best matches for a query are constrained to be entire database sequences, and (2) methods for subsequence matching, where the best matches for a query can be arbitrary subsequences of database sequences. Several well-known methods only address full sequence matching [14, 34, 40, 42], and cannot be easily used for efficient retrieval of subsequences.

Some methods transform subsequence matching to full sequence matching, by cutting database sequences into small pieces, and requiring each query to correspond to an entire such piece. One example is the query-by-humming system described in [43], where each database song is cut into smaller, disjoint pieces. Such approaches fail when the query corresponds to a database subsequence that is not stored as a single piece.

An indexing structure for unconstrained DP-based subsequence matching is proposed in [27]. However, as database sequences get longer, the time complexity for that method becomes similar to that of unoptimized DP-based matching. The method in [28] can handle such long database sequences; the key idea is to speed up DTW by reducing the length of both query and database sequences. The length is reduced by representing sequences as ordered lists of monotonically increasing or decreasing segments. By using monotonicity, that method is only applicable to 1-dimensional time series. A related method that can be used for multidimensional timeseries is proposed in [15]. In that method, time series are approximated by shorter sequences, obtained by replacing each constant-length part of the original sequence with the average value over that part.

The SPRING method for efficient subsequence matching under unconstrained DTW is proposed in [33]. In that method, optimal subsequence matches are identified by performing full sequence matching between the query and each database sequence. Subsequences are identified by prepending to each query a "null" symbol that matches any sequence prefix with zero cost. The complexity of that method is linear to both database size and query size.

The more powerful lower-bounding method LB_Keogh [14] was developed for efficient time series matching. The main idea is to use the warping constraint to create an envelope around the query sequence. Then, using a sliding window of size equal to the query, we can estimate a lower bound of the matching cost between the query and each possible subsequence. Since LB_Keogh gives a lower bound on the actual distance, this approach can be used to prune a large number of subsequences. For the subsequences that cannot be pruned, the exact dynamic programming algorithm is used to compute the distances and ultimately find the best match.

Music retrieval has been studied widely. Several algorithms have been developed and various music representations have been studied. Several probabilistic methods (HMMs) have been developed for music retrieval [3, 20, 35, 36]. However, they are computationally expensive due to the required training and very tough task of creating models to represent all kinds of music. An extended HMM architecture Factorial HMM has been proposed to model music [13], more specifically Bach's chorales. Factorial HMMs are based on a factored, distributed representation of the hidden state variable. Due to their complicated structure, inference and learning is intractable, and approximate learning is necessary. Although the model may be effective in capturing the statistical structure in the Bach's chorales, it is not built for any query processing as such.

Dynamic programming approaches seemed to be the most promising both in terms of accuracy and computational cost [12, 38, 18, 43].

3. SUBSEQUENCE MATCHING

We study three existing time series subsequence matching methods, Dynamic Time Warping, constrained Dynamic Time Warping, and SPRING.

3.1 Dynamic Time Warping

Dynamic time warping (DTW) identifies an optimal alignment between two time series, and computes the matching cost corresponding to that alignment. In DTW an individual element of one sequence can be matched with at least one and possibly more elements of the other sequence, thus allowing for each series to be stretched locally along the time axis.

Given two N-dimensional time series $Q = (Q_1, \ldots, Q_{|Q|})$ and $X = (X_1, \ldots, X_{|X|})$, the Dynamic Time Warping matching cost $D(Q, X)$ is defined recursively using a dynamic programming matrix D of size $(|Q|+1)\text{x}(|X|+1)$. A *null* element is added at the beginning of Q and X and has the property that it matches with another null element with a score of '0' and any other element with a score of ∞. Let D_{ij} denote the element at the i_{th} row and j_{th} column of D. Then, the dynamic time warping cost $D(Q, X)$ is defined as follows:

$$D_{0,0}(Q, X) = 0 \ . \tag{1}$$

$$D_{0,j}(Q, X) = \infty \ . \tag{2}$$

$$D_{i,0}(Q, X) = \infty \ . \tag{3}$$

$$D_{i,j}(Q, X) = L_p(Q_i, X_j) + \min \left\{ \begin{array}{l} D_{i,j-1}(Q, X) \\ D_{i-1,j}(Q, X) \\ D_{i-1,j-1}(Q, X) \end{array} \right. \tag{4}$$

$$\forall (i = 1, \ldots, |Q|; j = 1, \ldots, |X|) \ .$$

$$D(Q, X) = D_{|Q|,|X|}(Q, X) \ . \tag{5}$$

Notice that $L_p(Q_i, X_j)$ is the L_p norm difference of Q_i and X_j.

3.2 Constrained Dynamic Time Warping

Constrained DTW (cDTW) is obtained from DTW simply by placing an additional constraint, which narrows down the set of positions in one sequence that can be matched with a specific position in the other sequence. We consider only using the Sakoe-Chiba band [32] constraint where the lengths of the two time series sequences are the same. Given a warping width w, this constraint is defined as follows:

$$D_{i,j}(Q, X) = \infty \text{ if } |i - j| > w \ . \tag{6}$$

The term "Sakoe-Chiba band" is often used to characterize the set of (i, j) positions for which $D_{i,j}$ is not infinite. Notice that if $w = 0$, cDTW becomes the L_p distance. While a simple modification of DTW, cDTW has been shown to be significantly more efficient than DTW for full sequence matching [14], and to also produce more meaningful matching scores [31].

Given the above definitions, the subsequence match of Q in a database X is the subsequence $X_{\text{opt}} = (X_j, \ldots, X_{j+|Q|-1})$ that minimizes $D(Q, X_{\text{opt}})$. Similarly to other approaches for subsequence matching under cDTW, e.g., LB_Keogh [14], we require that the subsequence match have the same length as the query. A simple approach for finding the subsequence match of Q is the sliding-window approach: we simply compute the matching cost between Q and every subsequence of X that has length $|Q|$.

The sliding window approach is speeded up by the LB_Keogh [14] lower-bounding method, often by orders of magnitude, by computing an efficient lower bound of the matching cost, that can be used to reject many subsequences without computing the exact cDTW cost between Q and those subsequences.

3.3 SPRING

Both DTW and cDTW described above, need to use a sliding window in order to determine the optimal subsequence match of a query in a large database. A straightforward extension to the definition of DTW is to include an extra character at the beginning of the query sequence that has the property of matching with every database position with a score of '0'. This allows a warping path to start at database position and not always the first position (as in DTW). The recursive definitions for this extension should be adjusted accordingly as we need to store multiple warping paths and not only one.

The SPRING [33] algorithm uses the same recursive definitions as those used by DTW, with the only difference in Equations 2 and 5, which are now, respectively, changed to:

$$D_{0,j}(Q, X) = 0. \tag{7}$$

$$D(Q, X) = \min_{j=1,\ldots,|X|} \{D_{|Q|,j}(Q, X)\}. \tag{8}$$

This extra "sink" state allows a match to start at any position of the target sequence X. The computational time of SPRING is $O(|Q||X|)$. By defining $D_{0,j} = 0$, arbitrary prefixes of X are allowed to be skipped (i.e., matched with zero cost) before matching Q with the optimal subsequence in X. By defining $D(Q, X)$ to be the minimum $D_{|Q|,j}(Q, X)$, where $j = 1, \ldots, |X|$, we allow the best matching subsequence of X to end at any position j. Overall, this definition matches the entire Q with an optimal subsequence of X.

To speed up search, an embedding-based approximate method, EBSM [2], can be used. This method is properly designed for efficient subsequence matching under SPRING and can achieve significant speed-ups (of over an order of magnitude) with very low losses in accuracy.

4. EXPERIMENTAL EVALUATION

We evaluated the performance of DTW, cDTW, and SPRING on real music data.

4.1 Datasets and Queries

We created a database of 5,641 MIDI files that we gathered by performing an extensive search on the web, covering many music genres, such as blues, rock, pop, classical, jazz, country, and also national anthems and themes from movies and tv series. To obtain the representation of pitch intervals and IOIR, we did the following steps: first, for each MIDI file we extracted for all channels (a MIDI file consists of at most 16 channels) the highest pitch at every time click. Except for channel 10 which is used for drums, this channel was ignored. Then, we converted the tuples of pitch and time click to tuples of pitch intervals and IOIR. This procedure resulted in a music database consisting of a total of $10,749$ time series with length at least 200.

To get an overview of the performance of all three algorithms in a real-life setting, we use queries that are created by selecting random subsequences in the database and adding various amounts of noise. For evaluation, we use 200 subsequences selected uniformly at random over the $10,749$ tracks and within the track at a uniformly random starting point. We add noise at each query by replacing $x\%$ of the points by points selected randomly from the database. We test each query with 5, 10, 15, 20, 25, and 30% of noise. All experiments were performed on a quad-core Intel Core 2 Q9550 processor using Matlab with the Matlab Parallel Computing Toolbox and Matlab Compiler. Our implementations, data, and repeatability instructions are available at `http://www.hiit.fi/software/music`.

4.2 Evaluation

To evaluate the performance, we compared the three dynamic programming techniques in terms of the *ranking* of the queries. Consider a query Q and a database of n music songs, $\mathcal{X} = \{X_1, \ldots, X_n\}$, and assume that the true match of Q is sequence X_{match}. Given a subsequence matching method M, the *absolute rank* of Q is defined as the number of database sequences for which $M(Q, X_i) \leq M(Q, X_{match})$, where $M(Q, X_i)$ is the matching score of Q and X_i using method M. The *relative rank* of Q is the absolute rank divided by the number of tracks in the database. We are mainly interested in the performance of the methods in retrieving the correct song in the first K hits, where K is a small number, like 20 or 50.

Figure 2 shows the percentage of queries for which the true match is within the first 50 songs retrieved. We observe for 5% noise that the true match is within the first 10 matches in more than 95% of the cases. The performance degrades slowly as the amount of noise increases and even at a noise level of 25%, all three algorithms find the true match within the first ten matches in 4 out of 5 cases. We also find, in Figure 2, that the three algorithms perform similarly for this top-50 scenario. Also, if the song is not within the first 20 matches, then there is very little chance that it is within the first 50 matches at any noise level, which is indicated by a flat line in Figure 2.

In Figure 3 we find the full performance curves of the three algorithms for each noise level. We observe in all cases the curve of SPRING is, on average, lower than those of DTW and cDTW, indicating a lower performance. The difference between DTW and cDTW is very small in any case. So, cDTW can be considered superior, because it is by definition much faster to compute and, on top, effective lower bounding measures can be used.

In terms of retrieval runtime, SPRING is in turn faster than cDTW over a sliding window. We should mention that the top-K performance is more important for our specific application. Clearly, as also shown in Figure 2, all three methods are the same in terms of accuracy for $K = 50$, which is a realistic value for this application. Thus, in this setting, SPRING would be preferable to cDTW, as it can achieve the same accuracy at a lower runtime.

5. ASSISTIVE ENVIRONMENTS

In the context of event detection in assistive environments, the methods discussed could be used for multimedia and object detection and characterization. In such environments, noise is expected to be present in various measurements and representations. One common type of noise in time series is the repetition of the same event (i.e., note) or the occurrence of the same series of events but in different phase or frequency. For such types of noise, "warping distance"-based methods are highly applicable and recommended due to their ability to perform many-to-one mappings between the aligned time series. To avoid confusion, we should point out that other types of noise, such as additional events with relatively higher or lower values (i.e., notes with very high or low pitch values or durations with respect to the target sequence) are still a bottle-neck for the methods discussed in this paper. For such types of noise, LCSS-based approaches [5, 6, 37] would be more appropriate.

6. CONCLUSIONS

We studied the performance of three time series subsequence matching methods on the domain of music retrieval. Each music piece was represented by taking into account, for each note, both pitch and duration values. Our experiments show that DTW, cDTW, and SPRING have quite similar performance when the number of matches is relatively small, as in the top-K scenario. This suggests that in the case where query lengths are arbitrary, SPRING would be preferable due to its low computational cost, as opposed to DTW and cDTW. Significant speed-ups could be achieved by using LB_Keogh [14] and EBSM [2] for cDTW and SPRING respectively. However, in this work we only study the retrieval accuracy of the aforementioned dynamic programming methods.

7. REFERENCES

[1] T. Argyros and C. Ermopoulos. Efficient subsequence matching in time series databases under time and amplitude transformations. In *International Conference on Data Mining*, pages 481–484, 2003.

[2] V. Athitsos, P. Papapetrou, M. Potamias, G. Kollios, and D. Gunopulos. Approximate embedding-based subsequence matching of time series. In *ACM SIGMOD*, 2008.

[3] D. Bainbridge. MELDEX: A Web-based Melodic Locator Service. *Computing in Musicology*, 11:223–229, 1998.

[4] R. Bellman. The theory of dynamic programming. *Bull. Amer. Math. Soc.*, 60(6):503–515, 1954.

[5] L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithms. In *SPIREâĂŹ00: Proceedings of the Seventh International Symposium on String Processing Information Retrieval (SPIREâĂŹ00)*, pages 39–48.

[6] B. Bollobás, G. Das, D. Gunopulos, and H. Mannila. Time-series similarity problems and well-separated geometric sets. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 454–456. ACM New York, NY, USA, 1997.

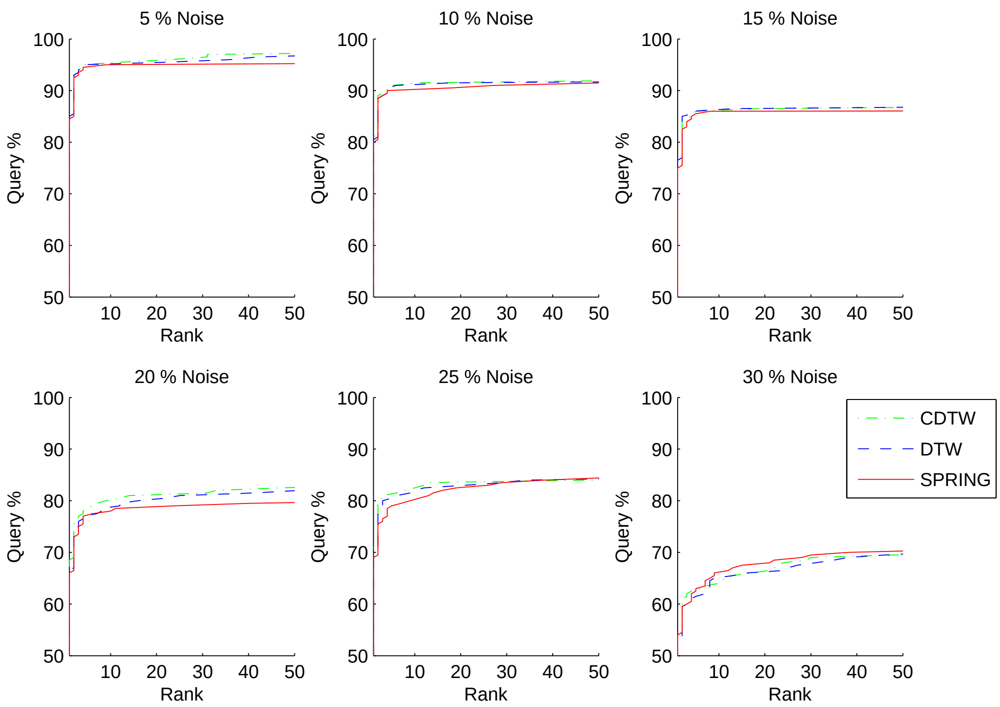

Figure 2: Retrieval accuracy curve for looking at 1 to 50 best matches for all three algorithms and different levels of noise. *Rank* corresponds to the absolute rank of the queries.

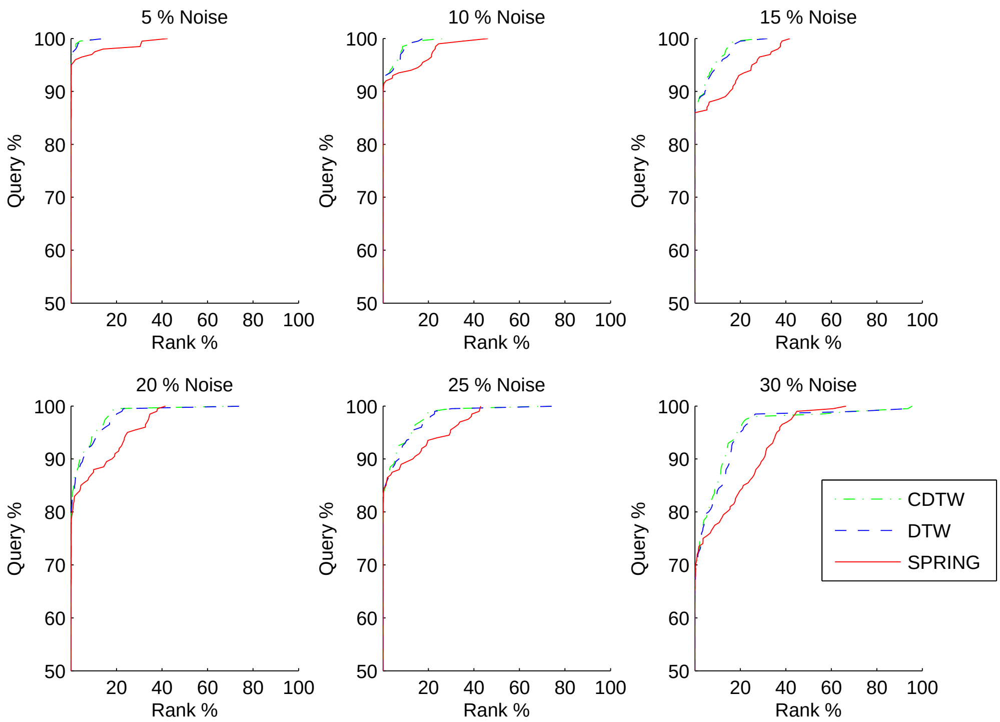

Figure 3: The full retrieval accuracy curve for all three algorithms and different levels of noise. *Rank %* corresponds to the relative rank of the queries.

[7] K.-P. Chan and A. W.-C. Fu. Efficient time series matching by wavelets. In *ICDE*, pages 126–133, 1999.
[8] L. Chen and R. T. Ng. On the marriage of lp-norms and edit distance. In *VLDB*, pages 792–803, 2004.
[9] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD Conference*, pages 491–502, 2005.
[10] R. Dannenberg and N. Hu. Understanding search performance in query-by-humming systems. In *Proc. ISMIR*, pages 232–237, 2004.
[11] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *ACM International Conference on Management of Data (SIGMOD)*, pages 419–429, 1994.
[12] W. Gao, G. Fang, D. Zhao, and Y. Chen. Transition movement models for large vocabulary continuous sign language recognition. In *Automatic Face and Gesture Recognition*, pages 553–558, 2004.
[13] Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. *Machine Learning*, 29:245–275, 1997.
[14] E. Keogh. Exact indexing of dynamic time warping. In *International Conference on Very Large Data Bases*, pages 406–417, 2002.
[15] E. Keogh and M. Pazzani. Scaling up dynamic time warping for data mining applications. In *Proc. of SIGKDD*, 2000.
[16] J. B. Kruskall and M. Liberman. The symmetric time warping algorithm: From continuous to discrete. In *Time Warps*. Addison-Wesley, 1983.
[17] L. Latecki, V. Megalooikonomou, Q. Wang, R. Lakämper, C. Ratanamahatana, and E. Keogh. Elastic partial matching of time series. In *PKDD*, pages 577–584, 2005.
[18] K. Lemström and E. Ukkonen. Including interval encoding into edit distance based music comparison and retrieval. In *Proc. AISB*, pages 53–60, 2000.
[19] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics*, 10(8):707–710, 1966.
[20] D. Mazzoni and R. Dannenberg. Melody matching directly from audio. In *2nd Annual International Symposium on Music Information Retrieval*, pages 17–18, 2001.
[21] Y. Moon, K. Whang, and W. Han. General match: a subsequence matching method in time-series databases based on generalized windows. In *ACM International Conference on Management of Data (SIGMOD)*, pages 382–393, 2002.
[22] Y. Moon, K. Whang, and W. Loh. Duality-based subsequence matching in time-series databases. In *IEEE International Conference on Data Engineering (ICDE)*, pages 263–272, 2001.
[23] P. Morguet and M. Lang. Spotting dynamic hand gestures in video image sequences using hidden Markov models. In *IEEE International Conference on Image Processing*, pages 193–197, 1998.
[24] M. Morse and J. Patel. An efficient and accurate method for evaluating time series similarity. In *SIGMOD Conference*, pages 569–580, 2007.
[25] R. Oka. Spotting method for classification of real world data. *The Computer Journal*, 41(8):559–565, July 1998.
[26] 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.
[27] S. Park, W. W. Chu, J. Yoon, and J. Won. Similarity search of time-warped subsequences via a suffix tree. *Information Systems*, 28(7), 2003.
[28] S. Park, S. Kim, and W. W. Chu. Segment-based approach for subsequence searches in sequence databases. In *Symposium on Applied Computing*, pages 248–252, 2001.
[29] S. Pauws. Cubyhum: A fully operational query by humming system. In *Proceedings of ISMIR*, pages 187–196, 2002.
[30] D. Rafiei and A. O. Mendelzon. Similarity-based queries for time series data. In *ACM International Conference on Management of Data (SIGMOD)*, pages 13–25, 1997.
[31] C. Ratanamahatana and E. J. Keogh. Three myths about dynamic time warping data mining. In *SIAM International Data Mining Conference*, 2005.
[32] H. Sakoe and S. Chiba. In *Transactions on ASSP*, volume 26, pages 43–49.
[33] Y. Sakurai, C. Faloutsos, and M. Yamamuro. Stream monitoring under the time warping distance. In *IEEE International Conference on Data Engineering (ICDE)*, 2007.
[34] Y. Sakurai, M. Yoshikawa, and C. Faloutsos. FTW: fast similarity search under the time warping distance. In *Principles of Database Systems (PODS)*, pages 326–337, 2005.
[35] J. Shifrin, B. Pardo, C. Meek, and W. Birmingham. HMM-based musical query retrieval. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 295–300. ACM New York, NY, USA, 2002.
[36] L. Smith, R. McNab, and I. Witten. Sequence-based melodic comparison: A dynamic programming approach. *Computing in Musicology*, 11:101–117, 1998.
[37] A. Uitdenbogerd and J. Zobel. Melodic matching techniques for large music databases. In *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, page 66. ACM, 1999.
[38] E. Unal, E. Chew, P. Georgiou, and S. Narayanan. Challenging uncertainty in query by humming systems: a fingerprinting approach. *IEEE Transactions on Audio Speech and Language Processing*, 16(2):359, 2008.
[39] M. Vlachos, D. Gunopulos, and G. Kollios. Discovering similar multidimensional trajectories. In *ICDE*, pages 673–684, 2002.
[40] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh. Indexing multi-dimensional time-series with support for multiple distance measures. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 216–225, 2003.
[41] H. Wu, B. Salzberg, G. C. Sharp, S. B. Jiang, H. Shirato, and D. R. Kaeli. Subsequence matching on structured time series data. In *ACM International*

Conference on Management of Data (SIGMOD), pages 682–693, 2005.

[42] B.-K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *IEEE International Conference on Data Engineering*, pages 201–208, 1998.

[43] Y. Zhu and D. Shasha. Warping indexes with envelope transforms for query by humming. In *ACM International Conference on Management of Data (SIGMOD)*, pages 181–192, 2003.