Query by Humming: a Time Series Database Approach

Yunyue Zhu
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, New York, NY 10012
yunyue@cs.nyu.edu

Dennis Shasha
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, New York, NY 10012
shasha@cs.nyu.edu

ABSTRACT

A Query by Humming system allows the user to find a song by humming part of the tune. No musical training is needed. Previous query by humming systems do not provide satisfactory results for various reasons. Some systems have low retrieval precision because they rely on melodic contour information from the humming tune, which in turn relies on the error-prone note segmentation process. Some systems yield better precision when matching the melody directly from audio, but they are slow because of their extensive use of Dynamic Time Warping (DTW). Our research approach improves both the retrieval precision and speed compared to previous approaches. We treat music as a time series and exploit and improve well-developed techniques from time series databases to index the music for fast similarity queries. We improve on existing DTW technique by introducing a general framework for indexing time series. The net result is high scalability. We confirm our claims through extensive experiments.

1. INTRODUCTION

You have a tune lingering in your head for many days, but you don't know where you heard this tune or which song it is from. You can't search it using an online search engine such as Google because you know nothing about the metadata of the tune. Often, a music store clerk acts as a musical search engine, interpreting tunes hummed by shoppers and directing them to an album. This works even for shoppers who can't hum very well, as the authors know from personal experience. Such a resource is very useful but hard to find.

A Query by Humming system is just such a resource. The user will hum a piece of tune into a microphone connected to a computer. The computer will search a database of tunes to find a list of melodies that are most similar to the user's "query". The user will then listen to this result to see if it is actually the tune that he had in mind. Sometimes the user won't get the tune he wanted because he hummed way off tune, the database does not contain that tune, or the computer is not intelligent enough to tell whether two tunes sound similar.

Query by humming is a particular case of "Query by Content" in multimedia databases. Most research into "query by humming" in the multimedia research community uses the notion of "Contour" information. Melodic contour is the sequence of relative differences in pitch between successive notes [9]. It has been shown to be a method that the listeners use to determine similarities between melodies. However, the inherent difficulty the contour method encounters is that there is no known algorithm that can reliably transcribe the user's humming to discrete notes. We discuss this point further below.

Recently, there has been work that matches a melody directly from audio [20, 12, 34]. This generally gives better query results because it is free from the error-prone note segmentation. However, because such work relies on Dynamic Time Warping (DTW), the performance is poor.

There have been reports of query by humming in the popular press [1]. A group of researchers from the Universities of London, Indiana and Massachusetts have built a system (technical details missing so far) that can allow a "query by one interpretation" of a classical piece and retrieve another interpretation of that same piece. Prospective applications are to query by humming.

The database community has been investigating similarity query in time series databases for a long time. These time series database techniques can be applied to build a fast and robust query by humming database system. This research will act as a bridge to close the gaps between the database, the multimedia, and information retrieval communities to provide a solution to query by content for music databases. Our time series approach makes indexing large music databases practical.

2. RELATED WORK

Most of the research in pre-existing query by humming systems uses pitch contour to match similar melodies [9, 6, 21, 30]. The user's humming is transcribed to a sequence of discrete notes and the contour information is extracted from the notes. This contour information is represented by a few letters. For example, ("U", "D", "S") represents that a note is above, below or the same as the previous one. Naturally, more letters can be introduced to get a finer measure of the contour. For example, “u” indicates that a note is slightly higher than the previous one while “U” indicates that it is much higher. The tunes in the databases are also represented by contour information. In this way, a piece of melody is represented by a string with a small alphabet. The edit distance can be use to measure the similarity between two melodies. Techniques for string matching such as “q-grams” can be used to speed up the similarity query. The advantage of using contours is that while most users can hum the contour correctly, they cannot hum the contour intervals correctly. But such work suffers from two serious problems.

- It has been shown that contour information alone is not enough to distinguish a large database of melodies. A typical query of six notes on a database of 2,697 tracks would result in about 330 tracks being returned [29].
- It is very hard to segment a user's humming into discrete notes. There are reliable algorithms to transcribe the user humming in each short time period to a specific pitch [28]. But no good algorithm is known to segment such a time series of pitches into discrete notes. The precision of the query system thus rests on an imprecise preprocessing stage.

To avoid the first problem, one can use longer query lengths and finer measures of contour intervals. But that requires too much of users, especially poor singers. The second problem is more fundamental. There are two solutions proposed in the literature.

1. The user is required to clearly hum the notes of the melody using only the syllable “ta”, “la” or “da” [19, 21]. But such an input method is very unnatural for users. Moreover, when there are tie notes in a tune, it is very hard for the user to articulate them correctly. Such a cumbersome job should be left to intelligent computer programs.
2. Some recent work [20, 12, 34] proposes to match the query directly from audio based on melody slope [34] or dynamic time warping [20, 12] to match the hum-query with the melodies in the music databases. The results reported using such methods are quite encouraging. Compared to note based methods, such direct methods generally have higher retrieval precision [20]. But this quality improvement comes at a price. Performance deteriorates to such an extent that [20] states “Perhaps the biggest criticism of our work is that it is clearly a brute-force approach and it is very slow.” No indices are used, which makes searching in a large music database unpractical.

The database community has been researching problems in similarity query for time series databases for many years. The techniques developed in the area might shed light on the query by humming problem. Agrawal et al. [2] utilized the Discrete Fourier Transform (DFT) to transform data from the time domain into the frequency domain and used a multidimensional index structure to index the first few DFT coefficients. The focus in their work was on whole sequence matching. This was generalized to allow subsequence matching [8, 22]. Rafiei and Mendelzon [26] improved this technique by allowing transformations, including shifting, scaling and moving average, on the time series before similarity queries. In addition to DFT [2, 26, 35], Discrete Wavelet Transform (DWT) [7, 31, 24], Singular Value Decomposition (SVD)[17], Piecewise Aggregate Approximation (PAA)[32, 14] and Adaptive Piecewise Constant Approximation [15] approaches have also been proposed for similarity searching.

Allowing Dynamic Time Warping (DTW) in time series similarity searching is very critical for a query by humming system. A point-by-point distance measure between time series is very likely to fail due to variations in the duration of notes. Berndt and Clifford [5] introduced the concept of DTW to the data mining community. They showed how to use Dynamic Programming to compute the DTW distance and demonstrated its application as a time series similarity measure. Yi et al. [33] were the first to investigate the DTW in very large databases. They proposed two techniques to speed up DTW in a pipeline fashion. The first technique is to use FastMap to index time series with the DTW distance measure. But this technique might result in false negatives. The second is a global lower-bounding technique for filtering out unlikely matches. In recent work, Keogh [14] proposed a technique for the exact indexing of DTW that guarantees no false negatives.

2.1 Our contributions

In this paper, we investigate the problem of indexing very large music databases, which allows efficient and effective query by humming. Our strategy and contributions are as follows.

- We treat both the melodies in the music databases and the user humming input as time series. Such an approach allows us to integrate many database indexing techniques into a query by humming system, improving the quality of such system over the traditional (contour) string databases approach.
- We propose a special searching technique called k-Local Dynamic Time Warping that are invariant to shifting, time scaling and local time warping. This makes the system robust and allows flexible user humming input, while attaining high speeds.
- We improve on the state-of-the-art indexing technique for time series databases allowing dynamic time warping due to [14] by 3 to 10 times in practice. We give a better lower-bound distance for dynamic time warping than the best lower-bound technique we know.
- As a general contribution, we introduce a framework for existing dimensionality reduction transformations, such as Discrete Fourier Transform (DFT) and Discrete Wavelet Transform (DWT), to allow time warping. This might have applications to video processing in the spirit of [14].

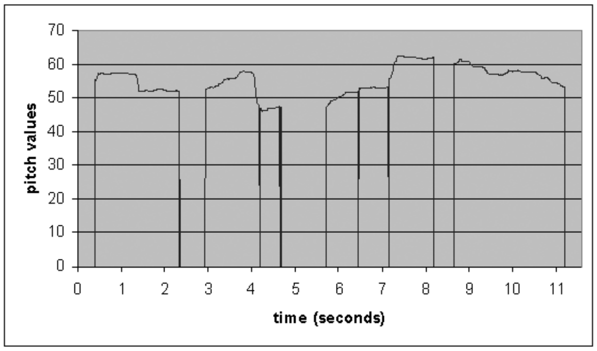

Figure 1: An example of a pitch time series. It is the tune of the beginning two phrases in the Beatles's song "Hey Jude" hummed by an amateur.

- The net effect is that our approach is scalable to very large music databases, as we demonstrate.

3. ARCHITECTURE OF QUERY BY HUMMING SYSTEM

A typical query by humming system includes three components:

- User humming: the input hum-query
- A database of music
- An index into the database for efficient retrieval of the hum-query

In this section, we will discuss these components in detail, with a focus on the indexing techniques.

3.1 User humming: the input hum-query

The user hums the query melody using a PC microphone with a single channel (mono). This acoustic input is segmented into frames of 10ms and each frame is resolved into a pitch using a pitch tracking algorithm [28]. This results in a time series of the pitches. Figure 1 shows an example of a pitch time series. It is the tune of the first two phrases in the Beatles's song "Hey Jude". The user may hum in any way he/she prefers. From the example of figure 1, we can see that it is very hard for the human being to mark the borders between notes. This is also the case for the computer. As mentioned above, we are not aware of any algorithm or product that can automatically segment notes with high accuracy. That is why our system avoids note segmentation.

3.2 A database of music

Our music database is made up of a collection of pieces of melodies. The input query will be matched with each piece of melody in the database. This works because most people hum melodic sections. This is a whole sequence matching approach in time series databases, as opposed to subsequence matching. Subsequence queries are generally much slower than whole sequence queries because the size of the potential candidate sequences is much larger.

A piece of melody is made up of a sequence of the tuples (Note, Duration). Because we use a monotone melody, there is only one note playing at each moment. A sequence of tuples $(N_1, d_1), (N_2, d_2), ..., (N_k, d_k)$ represents a piece of melody starting with note N_1 that lasts for d_1 time, followed by note N_2 that lasts for d_2 time... etc. Notice that we do not include the information of *rests* in the melody because amateur singers are notoriously bad in the timing of rests. In fact, we simply ignore the silent information in the user input humming and the candidate melodies in the database. Such a sequence of tuples can then be thought of as a time series in the following format:

$$\underbrace{N_1, N_1, \cdots, N_1}_{d_1}, \underbrace{N_2, N_2, \cdots, N_2}_{d_2}, ...$$

Figure 2 shows the melody of the beginning two phrases in the Beatles's "Hey Jude" and its time series representation.

3.3 Indexing databases for efficient humming query

If the user of the query by humming system were a good singer, we would just use the Euclidean distance between the time series to match the input pitch time series with the candidate time series in the database. The difficulty of query by humming comes from the fact that most users will not hum at the right pitch or tempo. The system should be flexible enough to allow typical inaccuracies in:

1. **Absolute pitch** Only about 1 in 10,000 people can get the absolute pitch right [25]. People of different genders, ages or even in different moods will hum the

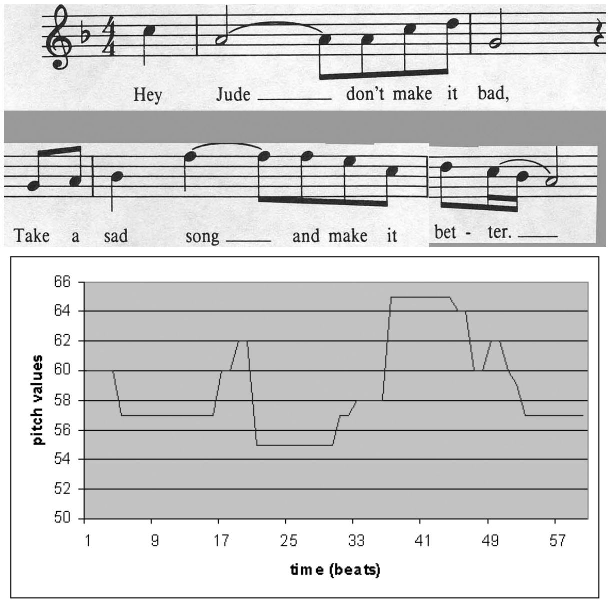

Figure 2: The sheet music of "Hey Jude" and its time series representation

same melody with very different pitches. The humming of most people will have more accuracy in the relative pitch, or the intervals. Our system allows the user to hum at different absolute pitches. To do this, we subtract from the time series their average pitches before the matching. This is a Shift-invariant technique for time series matching.

2. **Tempo** A song can be sung at different tempos and still sound quite the same. In practice, a melody will be hummed at a tempo that ranges from half to double the original tempo. However the tempo is usually more or less consistent, that is, when the tempo changes, the duration of each note changes proportionally. We can imagine this as a uniform stretching or squeezing of the time axis. In time series database research, this is called Time Scaling, or Uniform Time Warping.

3. **Relative pitch** The problem of variation in relative pitch for the average singer is less severe than that of the absolute pitch. But it is still not rare for notes to be sung a bit high or low. Suppose that the timing of each note is perfect, the distance between the query humming time series and a candidate time series can then be measured by the sum of the differences at each sample time moment. The smaller this distance is, the more similar a candidate melody is to the humming. So the problem of finding a similar melody is a Nearest Neighbors query.

4. **Local timing variation** Unfortunately, it is not realistic to require that the timing of each humming note is perfect. Using Dynamic Time Warping, we can make the matching process allow variations in tempo for each note. The idea is to stretch and squeeze the time axis locally to minimize the point-to-point distance of two time series. An important contribution of our paper is an efficient indexing scheme for local dynamic time warping.

In short, our time series approach first transforms time series to a "normal form" [10] to make the similarity measure invariant under shifting and time scaling. Figure 3 shows an example of the time series representations of the humming and the candidate music tune after they are transformed to their normal forms, that is, they have the same absolute pitch and tempo. The dynamic time warping distance between the normal forms of the time series will be used as the similarity measure.

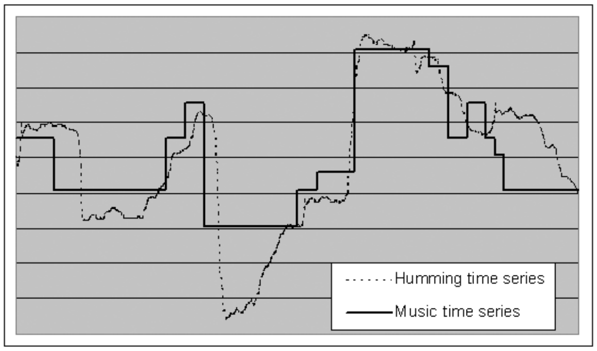

Figure 3: the time series representations of the humming and the candidate music tune after they are transformed to their normal forms

Table 1: list of symbols

D	Euclidean distance function
$\vec{x}$	time series
$\vec{x}^n$	time series of length n
x_i	the $i-$th element of $\vec{x}$
$U^w(\vec{x})$	$w-$upsample of a time series $\vec{x}$
$First(\vec{x})$	the first element of $\vec{x}$
$Rest(\vec{x})$	the rest time series of $\vec{x}$ without $First(\vec{x})$
$\vec{e}$	the envelope of a time series
$\vec{e^L}, \vec{e^U}$	the lower and upper part of envelope $\vec{e}$

To scale up to large databases, we must avoid a linear scan which examines the distance between the query time series and all the time series in the database. However, it is hard to index time series data because of their high dimension. To address the problem, the GEMINI framework [8] is used. The idea is to approximate the original time series and to reduce their dimensionality. Given a time series $\vec{x}^n$, a dimensionality reduction transform $\mathcal{T}$ will reduce it to a lower dimension $\vec{X}^N = \mathcal{T}(\vec{x}^n), N << n$. $\vec{X}^N$ is also called the feature vector of $\vec{x}^n$. After the time series are mapped to a lower dimensionality space, they can be indexed by a multidimensional index structure such as an R* tree [4] or a grid file [35]. To guarantee no false negatives in similarity queries, $\mathcal{T}$ must be lower-bounding, that is, the distance between time series under dimensionality reduction should lower-bound their original distance:

$$D(\mathcal{T}(\vec{x}), \mathcal{T}(\vec{y})) \leq D(\vec{x}, \vec{y}).$$

Popular dimensionality reduction transformations include the Fourier Transform, the Wavelet Transform, SVD and Piecewise Aggregate Approximation. We extend the GEMINI framework to the Dynamic Time Warping distance measure.

4. DYNAMIC TIME WARPING

We first summarize in table 1 a list of symbols used in the rest of paper. The Euclidean distance metric is the distance metric we use for time warping. Other distance metrics are also possible in our framework with some modifications.

The standard definition of Dynamic Time Warping distance is as follows [5, 33, 23]:

Definition 1. The Dynamic Time Warping distance between two time series $\vec{x}$,$\vec{y}$ is

$$D^2_{DTW}(\vec{x}, \vec{y}) = D^2(First(\vec{x}), First(\vec{y}))$$

$$+min\begin{cases} D^2_{DTW}(\vec{x}, Rest(\vec{y})) \\ D^2_{DTW}(Rest(\vec{x}), \vec{y}) \\ D^2_{DTW}(Rest(\vec{x}), Rest(\vec{y})) \end{cases}$$

The process of computing the DTW distance can be visualized as a string matching style dynamic program (figure

4). We construct a $n \times m$ matrix to align time series $\vec{x}^n$ and $\vec{y}^m$. The cell (i, j) corresponds to the alignment of the element x_i with y_j. A warping path, P, from cell $(0, 0)$ to $(n-1, m-1)$ corresponds to a particular alignment, element by element, between $\vec{x}^n$ and $\vec{y}^m$.

$$P = p_1, p_2, ..., p_L = (p_1^x, p_1^y), (p_2^x, p_2^y), ..., (p_L^x, p_L^y)$$

$$max(n, m) \leq L \leq n + m - 1$$

The distance between $\vec{x}^n$ and $\vec{y}^m$ on the warping path P is the distance between $x_{p_t^x}$ and $y_{p_t^y}, t = 1, 2, ..., L$. The constraints on the path P are:

- P must be monotonic :$p_t^x - p_{t-1}^x \geq 0$ and $p_t^y - p_{t-1}^y \geq 0$
- P must be continuous :$p_t^x - p_{t-1}^x \leq 1$ and $p_t^y - p_{t-1}^y \leq 1$

The number of possible warping paths grows exponentially with the length of the time series. The distance that is minimized over all paths is the Dynamic Time Warping distance. It can be computed using Dynamic Programming in $O(mn)$ time [5].

4.1 Uniform time warping

Uniform Time Warping (UTW) is a special case of DTW. The constraint imposed by UTW is that the warping path must be diagonal.

Definition 2. The Uniform Time Warping distance between two time series $\vec{x}^n, \vec{y}^m$ is

$$D_{UTW}^2(\vec{x}^n, \vec{y}^m) = \frac{\sum_{i=1}^{mn-1}(x_{\lfloor i/m \rfloor} - y_{\lfloor i/n \rfloor})^2}{mn}$$

For simplicity of notation, we stretch both time axis of $\vec{x}^n, \vec{y}^m$ to be mn. It is clear that if their greatest common divisor, $GCD(m, n) > 1$, the time axis can be stretched to only their least common multiple, $LCM(n, m)$.

Using the concept of upsampling, the definition of Uniform Time Warping can be simplified.

Definition 3. The $w-$upsampling of a time series $\vec{x}^n$ is

$$U^w(\vec{x}^n) = \vec{z}^{nw}$$

$$z_i = x_{\lfloor i/w \rfloor}, i = 0, 1, ..., nw - 1$$

Intuitively, $w-$upsampling repeats each value in a time series w times. The following lemma makes it possible to reduce the UTW distance to Euclidean distance.

Lemma 1.

$$D_{UTW}^2(\vec{x}^n, \vec{y}^m) = \frac{D^2(U^m(\vec{x}^n), U^n(\vec{y}^m))}{mn}$$

Uniform Time Warping is a generalization of Time Scaling. In time scaling, the length of one time series must be a multiple of the length of the other, while there is no such a restriction for UTW. Using UTW, we can compute the distance between time series of different lengths. The UTW normal form of a time series $\vec{x}^n$ is $U^w(\vec{x}^n)$, where nw is a predefined large number. It is not hard to adjust existing dimensionality reduction techniques to compute the UTW normal form of a time series.

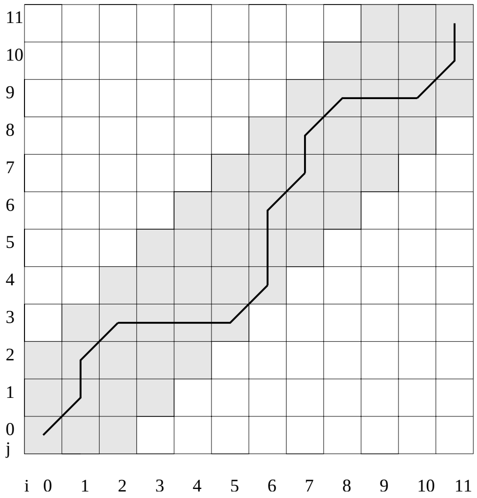

Figure 4: An example warping path with local constraint

4.2 Local dynamic time warping

The constraint imposed by UTW is too rigid, but it can be relaxed by Local Dynamic Time Warping (LDTW). Intuitively, humans will match two time series of different lengths as follows. First, the two time series are globally stretched to the same length. They are then compared locally point by point, with some warping within a small neighborhood in the time axis. Such a two-step transform can emulate traditional Dynamic Time Warping while avoiding some unintuitive results as well as speeding it up. Here is the definition of Local Dynamic Time Warping.

Definition 4. The $k-$Local Dynamic Time Warping distance between two time series $\vec{x}$,$\vec{y}$ is

$$D_{LDTW(k)}^2(\vec{x}, \vec{y}) = D_{constraint(k)}^2(First(\vec{x}), First(\vec{y}))$$

$$+min \begin{cases} D_{LDTW(k)}^2(\vec{x}, Rest(\vec{y})) \\ D_{LDTW(k)}^2(Rest(\vec{x}), \vec{y}) \\ D_{LDTW(k)}^2(Rest(\vec{x}), Rest(\vec{y})) \end{cases}$$

$$D_{constraint(k)}^2(x_i, y_j) = \begin{cases} D^2(x_i, y_j) & if|i-j| \leq k \\ \infty & if|i-j| > k \end{cases}$$

Figure 4 shows a warping path for $D_{LDTW(2)}$in a time warping grid. The possible paths are constrained to be within the shadow, which is a beam of width $5(= 2*2+1)$ along the diagonal path. The warping width is defined as $\delta = \frac{2k+1}{n}$. Such a constraint is also known as a Sakoe-Chiba Band. Other similar constraints are also discussed in [14]. It can be shown that the complexity of computing $k-$Local Dynamic Time Warping Distance is $O(kn)$ using dynamic programming.

Note that in the work [14], the definition of the DTW is actually LDTW. By combining UTW and LDTW together, we define a more general DTW distance:

Definition 5. The Dynamic Time Warping distance between two time series is the LDTW distance between their UTW normal forms.

In other words, it is the LDTW distance between two time series after they are both upsampled to be of the same length. In a slight abuse of notation, we will not distinguish LDTW and DTW in the remaining of the paper, and we assume the distance of LDTW is computed after the UTW transform had been performed on the time series.

4.3 Lower-bounding technique and indexing scheme

The Local Dynamic Time Warping distance makes it possible to lower-bound the distance locally. Because such a local lower-bound is tighter than the global lower-bounding for DTW [33], it produces fewer false positives.

To simplify our notation, we will first introduce the concept of envelope for time series.

Definition 6. The $k-$Envelope of a time series $\vec{x} = x_i, i = 1, ..., n$ is

$$Env_k(\vec{x}) = (Env_k^L(\vec{x}); Env_k^U(\vec{x})).$$

$Env_k^L(\vec{x})$ and $Env_k^U(\vec{x})$ are the upper and lower envelope of $\vec{x}$ respectively:

$$Env_k^L(\vec{x}) = x_i^L, i = 1, ..., n; x_i^L = min_{-k \leq j \leq k}(x_{i+j})$$

$$Env_k^U(\vec{x}) = x_i^U, i = 1, ..., n; x_i^U = max_{-k \leq j \leq k}(x_{i+j})$$

The distance between a time series and an envelope is defined naturally as follows.

Definition 7. The distance between a time series $\vec{x}$ and an envelope $\vec{e}$ is

$$D(\vec{x}, \vec{e}) = \min_{\vec{z} \in \vec{e}} D(\vec{x}, \vec{z})$$

We use $\vec{z}^n \in \vec{e}$ to denote that $e_i^L \leq z_i \leq e_i^U, i = 1, 2, ..., n$. So the value at each point can be any one in the range.

Keogh [14] proved that the distance between a time series and the envelope of another time series lower-bounds the true DTW distance.

Lemma 2. [14]

$$D(\vec{x}^n, Env_k(\vec{y}^n)) \leq D_{DTW(k)}(\vec{x}^n, \vec{y}^n)$$

To index the time series in the GEMINI framework, one needs to perform dimensionality reduction on the time series and its envelope. Piecewise Aggregate Approximation (PAA) is used in [14]. The PAA of the time series $\vec{x}^n$ is $\vec{X}^N$, $N << n$, where

$$X_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} x_j, i = 1, 2, ..., N.$$

That is, the n dimensional time series $\vec{x}^n$ is reduced to dimension N by taking averages in N consecutive equal sized "frames". The PAA reduction of the envelopes using Keogh's method is as follows. Let $(\vec{L}^N; \vec{U}^N)$ be the PAA reduction of an envelope $(\vec{l}^n; \vec{u}^n)$,

$$L_i = \min(l_{\frac{n}{N}(i-1)+1}, ..., l_{\frac{n}{N}i}),$$

$$U_i = \max(u_{\frac{n}{N}(i-1)+1}, ..., u_{\frac{n}{N}i}), i = 1, 2, ..., N.$$

The PAA of an envelope is just the piecewise constant function, which bounds but does not intersect the envelope.

We introduce a new PAA reduction as follows.

$$L_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} l_j, \qquad U_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} u_j.$$

$$i = 1, 2, ..., N$$

In our method, $\vec{U}$ and $\vec{L}$ are also piecewise constant functions, but each piece is the average of the upper or lower envelope during that time period. Figure 5-a shows a time series, its bounding envelope and the PAA reduction of the envelope using Keogh's method. Figure 5-b shows the same time series, its bounding envelops and the PAA reduction of the envelope using our method. We can see clearly that the bounds in figure 5-b are tighter than that in figure 5-a and it is straightforward to prove that this is always the case for any time series. We will show that our bounds can still guarantee to lower-bound the real DTW distance.

Before we prove that our PAA transform can provide a lower-bound for DTW, we will first introduce a framework to index time series under the DTW distance that allows general dimensionality reduction techniques. This is a principle contribution of the paper. We define the container property of a dimensionality reduction for an envelope as follows.

Definition 8. We say a transformation $\mathcal{T}$ for an envelope $\vec{e}$ is "container-invariant" if

$$\forall \vec{x}^n \ if \ \vec{x}^n \in \vec{e} \ then \ \mathcal{T}(\vec{x}^n) \in \mathcal{T}(\vec{e})$$

Just as a transform of time series that is lower-bounding can guarantee no false negatives for Euclidean distance, a transform of envelopes that is container-invariant can guarantee no false negatives for DTW distance.

Theorem 1. If a transformation $\mathcal{T}$ is container-invariant and lower-bounding then

$$D(\mathcal{T}(\vec{x}), \mathcal{T}(Env_k(\vec{y}))) \leq D_{DTW(k)}(\vec{x}, \vec{y})$$

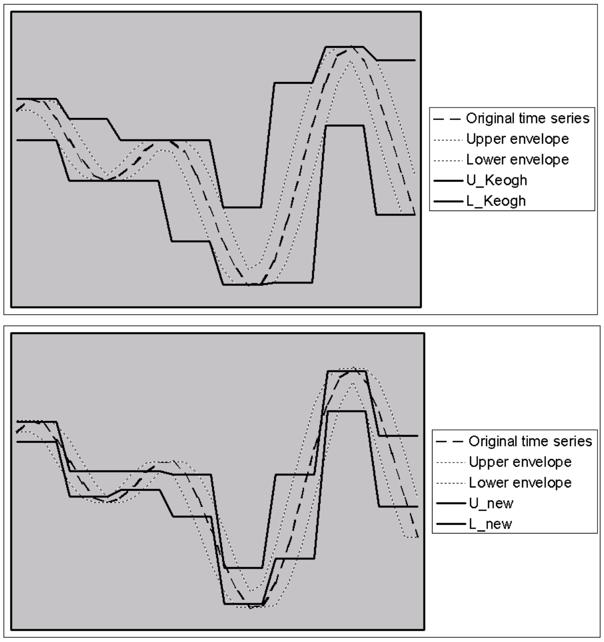

Figure 5: The PAA for the envelope of a time series using (a)Keogh's method(top) and (b)Our method(bottom).

PROOF. $\mathcal{T}$ is container-invariant,

$$\therefore \forall \vec{z}^n \; if \; \vec{z}^n \in \vec{e} \; then \; \mathcal{T}(\vec{z}^n) \in \mathcal{T}(\vec{e})$$

$$\therefore \{\vec{z} | \vec{z} \in Env_k(\vec{y})\} \subseteq \{\vec{z} | \mathcal{T}(\vec{z}) \in \mathcal{T}(Env_k(\vec{y}))\}$$

$$\therefore \min_{\{\vec{z}|\mathcal{T}(\vec{z})\in\mathcal{T}(Env_k(\vec{y}))\}} D(\vec{x},\vec{z}) \le \min_{\{\vec{z}|\vec{z}\in Env_k(\vec{y})\}} D(\vec{x},\vec{z})$$

$\mathcal{T}$ is lower-bounding,

$$\therefore D(\mathcal{T}(\vec{x}),\mathcal{T}(\vec{z})) \le D(\vec{x},\vec{z})$$

$$\therefore \min_{\{\vec{z}|\mathcal{T}(\vec{z})\in\mathcal{T}(Env_k(\vec{y}))\}} D(\mathcal{T}(\vec{x}),\mathcal{T}(\vec{z})) \le \min_{\{\vec{z}|\mathcal{T}(\vec{z})\in\mathcal{T}(Env_k(\vec{y}))\}} D(\vec{x},\vec{z})$$

$$\therefore \min_{\{\vec{z}|\mathcal{T}(\vec{z})\in\mathcal{T}(Env_k(\vec{y}))\}} D(\mathcal{T}(\vec{z}),\mathcal{T}(\vec{z})) \le \min_{\{\vec{z}|\vec{z}\in Env_k(\vec{y})\}} D(\vec{x},\vec{z})$$

By the definition of distance between time series and envelope,

$$D(\mathcal{T}(\vec{x}),\mathcal{T}(Env_k(\vec{y}))) \le D(\vec{x}, Env_k(\vec{y}))$$

From lemma 2,

$$D(\mathcal{T}(\vec{x}),\mathcal{T}(Env_k(\vec{y}))) \le D_{DTW(k)} D(\vec{x},\vec{y})$$

□

Using the concept of container-invariant, we can design the transform for the envelope based on PAA, DWT, SVD and DFT. All these dimensionality reduction transforms are linear transforms[1], that is,

$$\vec{X}^N = \mathcal{T}(\vec{x}^n), X_j = \sum_{i=1}^{n} a_{ij} x_i, j = 1, 2, ..., N.$$

We can extend such linear transforms for time series to transforms for time series envelopes, and at the same time guarantee they are container-invariant.

Lemma 3. Let transform $\mathcal{T}$ be a linear transform, $\vec{X}^N = \mathcal{T}(\vec{x}^n), X_j = \sum_{i=1}^{n} a_{ij} x_i, j = 1, 2, ..., N$, and the transform $\mathcal{T}$ on envelope $\vec{e}$ is as follows,

$$E = (\vec{E}^L, \vec{E}^U) = \mathcal{T}(\vec{e}^L, \vec{e}^U)$$

$$E_j^U = \sum_{i=1}^{n} (a_{ij} e_i^U \tau(a_{ij}) + a_{ij} e_i^L (1 - \tau(a_{ij})))$$

$$E_j^L = \sum_{i=1}^{n} (a_{ij} e_i^L \tau(a_{ij}) + a_{ij} e_i^U (1 - \tau(a_{ij})))$$

$$j = 1, 2, ..., N$$

where τ is the sign function:

$$\tau(x) = \begin{cases} 1 & x \ge 0 \\ 0 & x < 0 \end{cases}$$

Then transform $\mathcal{T}$ is container-invariant.

PROOF.

$$\forall \vec{x}^n \; if \; \vec{x}^n \in \vec{e} \; then \; e_i^L \le x_i \le e_i^U, i = 1, 2, ..., n$$

$$Therefore \; E_i^L \le X_j \le E_i^U, j = 1, 2, ..., N$$

$$\therefore \mathcal{T}(\vec{x}^n) \in \mathcal{T}(\vec{e})$$

□

A transform of an envelope is still an envelope, which we call the envelope in the feature space. In the special case when the envelope equals the time series, the transform of the envelope becomes the transform of the time series. Because PAA is a linear transform, and our proposed PAA transform for envelopes is deduced from the lemma above, it is container-invariant. PAA also has the nice property that all the coefficients of linear transformation for PAA, unlike DFT and SVD, are positive. In this case, the upper envelope in feature space is just the PAA transform of the upper envelope, so is the lower envelope. For other transform like DFT and SVD, the upper envelope in feature space is the transform of the combination of the upper and the lower envelope. So the envelopes in the PAA feature space are tighter then those for DFT and SVD in general. Our experiments also confirm this.

Now we are ready to describe the strategy for time series database query with DTW. An ϵ-range similarity query in a time series database is to find all the time series whose distances with the query are less than ϵ. It includes the following step.

1. For each time series $\vec{x}^n$ in the database, compute its feature vector $\vec{X}^N$.
2. Build an N-dimensional index structure on $\vec{X}^N$.

[1] DFT is a linear transform because the real and image part of a DFT coefficient are still a linear combination of the original time series.

3. For a time series query $\vec{q}^n$, compute its envelope $\vec{e}^n$ and $\vec{E}^N = \mathcal{T}(\vec{e}^n)$.

4. Make an ϵ-range query of $\vec{E}^N$ on the index structure, and return a set of time series S.

5. Filter out the false positives in S using their true DTW distances with $\vec{q}^n$. We can guarantee no false negatives from theorem 1.

Similarly, a k-nearest neighbors query can be built on top of such a range query [18, 27]. For existing time series databases indexed by DFT, DWT, PAA, SVD, etc., we can add Dynamic Time Warping support without rebuilding indices. This works because our framework allows all the linear transforms and adding the DTW support requires changes only to the time series query.

5. EXPERIMENTS

Our experiments are divided into three parts. First we will run experiments to evaluate the quality of our query by humming system using the time series database approach. Comparisons with traditional contour based method will be made. We will also test the efficiency of our DTW indexing scheme comparing to the state-of-the-art technique. Finally we will test the scalability of our system.

5.1 Quality of the query by humming system

We collected 50 of the most popular Beatles's songs by manual entry. These songs are further segmented to 1000 short melodies. Each melody contains 15 to 30 notes. We asked people with different musical skills to hum for the system.

To evaluate the quality of the query by humming system, first we compared it with the note contour-based method. We will use the hum queries of better singers in this experiment, because for hum queries of poor quality it is hard for even a human being to recognize the target song. For the note contour-based method, we need to transcribe the user humming into discrete notes first. For lack of a reliable note-segmentation algorithm, we used the best commercial software we could find, *AKoff Music Composer*[3], to transcribe notes. We also applied a standard algorithm [28] to transcribe the user hum query to a sequence of pitches, and used the silence information between pitches to segment notes. For the contour-based method, we report the better result based on these two note-segmentation processes. For the 20 pieces of hum queries by better singers, we searched the database to find their ranks using the contour-based approach and the time series approach. The result is shown in table 2. We can see that our time series approach beats the contour approach clearly. We are not claiming that using contour information for music matching is bad. However until a reliable note-segmentation algorithm is developed, such an approach is based on dubious input. If for example, the query input were by piano instead of human voice so that each individual note is clearly separated, we would expect the contour-base approach to have good quality too.

Table 2: The number of melodies correctly retrieved using different approaches

Rank	Time series Approach	Contour Approach
1	16	2
2-3	2	0
4-5	2	0
6-10	0	4
10-	0	14

Table 3: The number of melodies correctly retrieved by poor singers using different warping widths

Rank	$\delta = 0.05$	$\delta = 0.1$	$\delta = 0.2$
1	2	4	2
2-3	2	3	5
4-5	4	5	7
6-10	3	5	4
10-	9	3	2

We tested our system with some hum queries of poor quality, for example, by one of the authors. We define a melody to be *perfectly matched* if it is the intended target melody of the hummer and its rank is 1. The number of melodies perfectly matched is low. Still the result is quite encouraging. We noticed that the warping width can be adjusted to tune the query results. It is hard for a poor hummer to keep the right duration for each note of the melody. Allowing larger warping widths will give the hummers more flexibility in the duration of the notes. For the 20 hum queries by poor singers, we searched the database to find their ranks using DTW with different warping width. The result is reported in table 3. We can see that more queries return in the top 10 matches when the warping width is increased from 0.05 to 0.1. But this tendency disappears when the warping width is increased to 0.2, because it is unlikely for a hummer to sing way off tempo. When the warping width is too large, some melodies that are very different will have a small DTW distance too. A warping width of 1 for local DTW degenerates to global DTW. Larger warping widths also slow down the processing.

5.2 Experiments for indexing DTW

Having shown that our time series approach for query by humming system has superior quality over the traditional contour approach, we will also demonstrate that it is very efficient. Unlike [20], the performance of the time series approach does not suffer from the extensive use of DTW. The scalability of our system comes from our proposed technique for indexing DTW. We will compare our DTW indexing technique with the best existing DTW indexing method [14]. There is an increasing awareness to use a benchmark approach in time series database experiments to guard against implementation bias and data bias. In the spirit of the work [16, 14], we took such an approach to conduct our experiments. To avoid data bias, we conducted our experiments on a wide range of time series datasets [13] that cover disciplines including finance, medicine, industry, astronomy and music. We also measured the results in an implementation free fashion to avoid bias in implementation.

We define the tightness of the lower bound for DTW distance as follows.

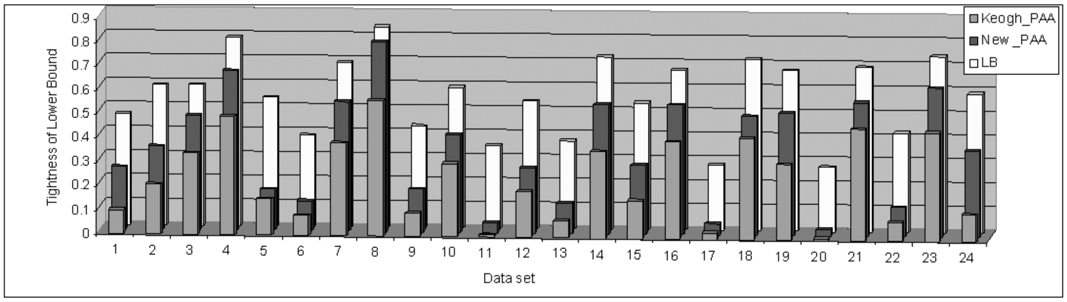

Figure 6: The mean value of the tightness of lower bound, using LB, New_PAA and Keogh_PAA for different time series data sets. The data sets are 1.Sunspot; 2.Power; 3.Spot Exrates; 4.Shuttle; 5.Water; 6. Chaotic; 7.Streamgen; 8.Ocean; 9.Tide; 10.CSTR; 11.Winding; 12.Dryer2; 13.Ph Data; 14.Power Plant; 15.Balleam; 16.Standard &Poor; 17.Soil Temp; 18.Wool; 19.Infrasound; 20.EEG; 21.Koski EEG; 22.Buoy Sensor; 23.Burst; 24.Randow walk

$$T = \frac{\textit{Lower Bound of DTW distance based on reduced dimension}}{\textit{True DTW distance}}$$

T is in the range of [0,1]. Larger T gives a tighter bound. Note that the definition here is different from that in the work [14]. In [14], Keogh has shown convincingly that lower-bounding using the envelope is much tighter than global bounding as reported in [33]. But such an envelope uses much more information than global lower-bounding. For a time series of size n, its envelope is represented by $2n$ values. By contrast, the global lower-bounding technique can be seen as using the minimum and maximum value of a time series as the envelope of a time series. So the global bounding is represented by only 2 values. To test the efficiency of dimensionality reduction under DTW, we modify the definition of T slightly, i.e., the lower-bound is based on reduced dimension. We compared three methods: LB is the lower-bound using the envelope (without dimensionality reduction and therefore without the possibility of indexing); Keogh_PAA is the lower-bound using PAA transformation proposed by Keogh [14] and New_PAA is our proposed PAA lower-bound. We chose each sequence to be of length $n = 256$ and a warping width to be 0.1. The dimension was reduced from 256 to 4 using PAA. We selected 50 time series randomly from each dataset and subtracted the mean from each time series. We computed the tightness of the lower bound of the distances between each pair of the 50 time series. The average tightness of lower bound for each dataset using the three methods are reported in figure 6.

From the figure, we can see that the method LB has the best T for each dataset. This is not a surprise, because the method LB uses much more information than Keogh_PAA and New_PAA. We include it here as a sanity check. LB will be used as a second filter after the indexing scheme, Keogh_PAA or New_PAA, returns a superset of answer. Using the same number of values, New_PAA is always better than Keogh_PAA. That comes from the fact that the estimations of DTW using New_PAA are always closer to the true DTW distance than the estimations using Keogh_PAA. The tightness of lower bound of New_PAA is approximately 2 times that of Keogh_PAA on average for all datasets.

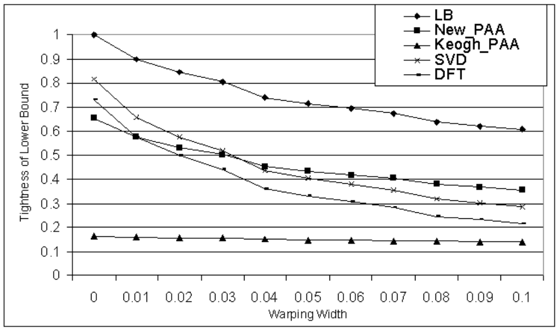

Figure 7: The mean value of the tightness of lower bound changes with the warping widths, using LB, New_PAA, Keogh_PAA, SVD and DFT for the random walk time series data set.

One of the contributions of this paper is a framework allowing DTW indexing using different dimensionality reduction methods in addition to PAA. The performance of competing dimensionality reduction methods under the Euclidean distance measure is very data-dependent, none of them is known to be the best in all cases. Allowing different dimensionality reduction methods to be extended to the case of the DTW distance measure will provide the users with more flexibility in choosing the right method for a particular application. We tested the tightness of lower bounds for DTW indexing using dimensionality reduction methods including PAA, DFT and SVD. For brevity, we will only report the results for the random walk data in figure 7, because the random walk data are the most studied dataset of time series indexing and are very homogeneous. We varied the warping widths from 0 to 0.1. Each T value is the average of 500 experiments. Again LB is always the tightest lower-bound because no dimensionality reduction is performed. In the case of 0 warping width, the DTW distance is the same as the Euclidean distance. Since SVD is the optimal dimensionality reduction method for Euclidean distance, the lower-bound using SVD is tighter than any other dimensionality reduction methods. The performance of DFT and PAA for the Euclidean distance measure is similar, which confirms other research [31, 15]. For all the warping widths,

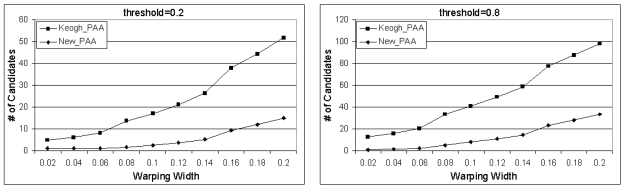

Figure 8: The number of candidates to be retrieved with different query thresholds for the Beatles's melody database

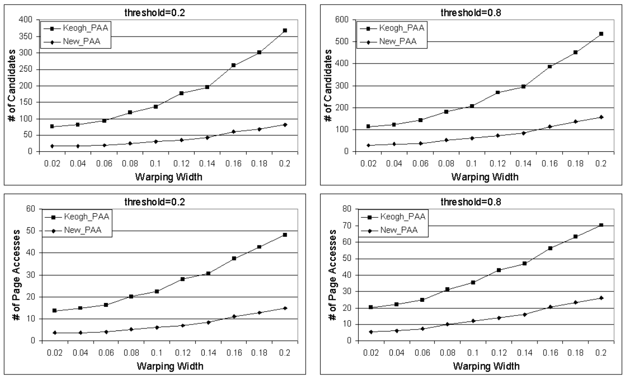

Figure 9: Performance comparisons with different query thresholds for a large music database

New_PAA is always better that Keogh_PAA as we would expect. New_PAA also beats DFT and SVD as the warping widths increase. The reason is that all the linear transformation coefficients for PAA are positive, as we mentioned before.

5.3 Scalability testing

The tightness of the lower bound is a good indicator of the performance of a DTW indexing scheme. A tighter lower-bound means that fewer candidates need to be retrieved for further examination in a particular query. That will increase the precision of retrieval at no cost to recall. Higher precision of retrieval implies lower CPU cost and IO cost at the same time, because we need to access fewer pages to retrieve candidate time series and to perform fewer exact Dynamic Time Warping computations. We will use the number of candidates retrieved and the number of page accesses as the implementation-bias free measures for the CPU and IO cost.

First we conducted experiments on our small music database of the Beatles's songs. Figure 8 shows the average number candidates to be retrieved for queries with different selectivity. The range queries have range $n\epsilon$, n is the length of the time series and the thresholds ϵ take the values of 0.2 and 0.8. From the figure, we can see that as the warping widths get larger, the number of candidates retrieved increases, because the lower-bounds get looser for larger warping widths. Our approach (New_PAA) is up to 10 times better than Keogh_PAA.

To test the scalability of our system, we need to use larger datasets. The first database we tested is a music database. We extracted notes from the melody channel of MIDI files we collected from the Internet and transformed them to our time series representation. There are $35,000$ time series in the database. The second database contains $50,000$ random walk data time series. Each time series has a length of 128 and is indexed by its 8 reduced dimensions using an R* tree [4] implemented in LibGist [11]. Each result we report is averaged over 500 experiments. Figure 9 shows the performance comparisons for the music database. We can see that the number of page accesses is proportional to the number of candidates retrieved for all the methods and thresholds. In a Pentium 4 PC, NEW_PAA took from 1 second for the smallest warping width to 10 seconds for the largest warping width. As the warping width increases, the number of candidates retrieved increases significantly using the Keogh_PAA method while it increases less for New_PAA. This is also true for other time series datasets. Figure 10 shows the performance comparisons for the random walk database. Similar performance advantages of our method hold for the random walk data too.

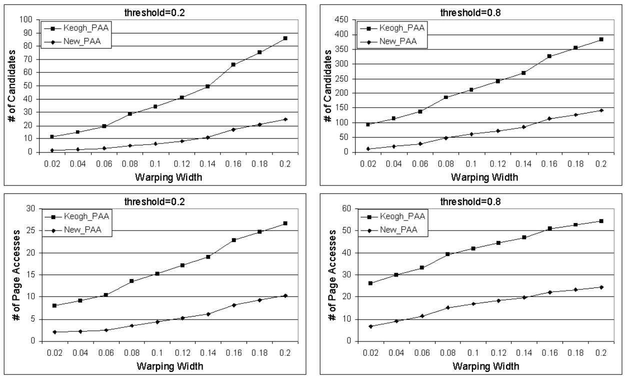

Figure 10: Performance comparisons with different query thresholds for a large random walk database

6. CONCLUSIONS

In this paper, we present a time series database approach for query by humming that gives high precision and is fast and scalable.

We have also introduced a general framework for indexing time series databases using the Dynamic Time Warping distance measure. This framework explains how to adapt existing time series indexing schemes for the Euclidean distance measure to the DTW distance measure. We prove that such an indexing scheme guarantees no false negatives. Within this framework, our PAA transform for DTW is consistently better than the previous reported PAA transform. Extensive experiments showed that the improvement is by a factor between 3 and 10.

We have also implemented a query by humming system. Preliminary testing of the system on real people (OK, our friends) gave good performance and high satisfaction. Some even improved their singing as a result. The system's potential applications to entertainment and education are *fortissimo.*

7. REFERENCES

[1] Computers can now recognise even mangled music. In *The Economist, Oct 5th 2002.*

[2] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient Similarity Search In Sequence Databases. In D. Lomet, editor, *Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms (FODO)*, pages 69–84, Chicago, Illinois, 1993. Springer Verlag.

[3] AKoff_Sound_Labs. Akoff music composer version 2.0,http://www.akoff.com/music-composer.html, 2000.

[4] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, May 23-25, 1990*, pages 322–331, 1990.

[5] D. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *Advances in Knowledge Discovery and Data Mining*, pages 229–248. AAAI/MIT, 1994.

[6] S. G. Blackburn and D. C. DeRoure. A tool for content based navigation of music. In *ACM Multimedia 98*, pages 361–368, 1998.

[7] K.-P. Chan and A. W.-C. Fu. Efficient time series matching by wavelets. In *Proceedings of the 15th International Conference on Data Engineering, Sydney, Australia*, pages 126–133, 1999.

[8] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 419–429, 1994.

[9] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith. Query by humming: Musical information retrieval in an audio database. In *ACM Multimedia 1995*, pages 231–236, 1995.

[10] D. Q. Goldin and P. C. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In *Proceedings of the 1st International Conference on Principles and Practice of Constraint Programming (CP'95)*, 1995.

[11] J. M. Hellerstein, J. F. Naughton, and A. Pfeffer. Generalized search trees for database systems. In U. Dayal, P. M. D. Gray, and S. Nishio, editors, *Proc. 21st Int. Conf. Very Large Data Bases, VLDB*, pages 562–573. Morgan Kaufmann, 11–15 1995.

[12] J.-S. R. Jang and H.-R. Lee. Hierarchical filtering method for content-based music retrieval via acoustic input. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 401–410. ACM Press, 2001.

[13] E. Keogh and T. Folias. The ucr time series data mining archive[http://www.cs.ucr.edu/ eamonn/tsdma/index.html],riverside ca. university of california - computer science and engineering department, 2002.

[14] E. J. Keogh. Exact indexing of dynamic time warping. In *VLDB 2002,Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*, pages 406–417, 2002.

[15] E. J. Keogh, K. Chakrabarti, S. Mehrotra, and M. J. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. In *Proc. ACM SIGMOD International Conf. on Management of Data*, 2001.

[16] E. J. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. In *the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,July 23 - 26, 2002. Edmonton, Alberta, Canada*, pages 102–111, 2002.

[17] F. Korn, H. V. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pages 289–300, 1997.

[18] F. Korn, N. Sidiropoulos, C. Faloutsos, E. Siegel, and Z. Protopapas. Fast nearest neighbor search in medical image databases. In *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 215–226, 1996.

[19] N. Kosugi, Y. Nishihara, T. Sakata, M. Yamamuro, and K. Kushima. A practical query-by-humming system for a large music database. In *ACM Multimedia 2000*, pages 333–342, 2000.

[20] D. Mazzoni and R. B. Dannenberg. Melody matching directly from audio. In *2nd Annual International Symposium on Music Information Retrieval,Bloomington, Indiana, USA*, 2001.

[21] R. J. McNab, L. A. Smith, D. Bainbridge, and I. H. Witten. The new zealand digital library melody index. In *D-Lib Magazine*, 1997.

[22] Y.-S. Moon, K.-Y. Whang, and W.-S. Han. Generalmatch: A subsequence matching method in time-series databases based on generalized windows. In *SIGMOD 2002,Madison,Wisconsin, USA, 3-6 June 2002.*, 2002.

[23] S. Park, W. W. Chu, J. Yoon, and C. Hsu. Fast retrieval of similar sub-sequences under time warping. In *ICDE*, pages 23–32, 2000.

[24] I. Popivanov and R. J. Miller. Similarity search over time series data using wavelets. In *ICDE*, 2002.

[25] J. Profita and T.G.Bidder. Perfect pitch. In *American Journal of Medical Genetics*, pages 763–771, 1988.

[26] D. Rafiei and A. Mendelzon. Similarity-based queries for time series data. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 13–25, 1997.

[27] T. Seidl and H.-P. Kriegel. Optimal multi-step k-nearest neighbor search. In L. M. Haas and A. Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 154–165, 1998.

[28] T. Tolonen and M. Karjalainen. A computationally efficient multi-pitch analysis model. *IEEE Transactions on Speech and Audio Processing*, 2000.

[29] A. Uitdenbgerd and J. Zobel. Manipulation of music for melody matching. In *ACM Multimedia 98*, pages 235–240, 1998.

[30] A. Uitdenbgerd and J. Zobel. Melodic matching techniques for large music databases. In *ACM Multimedia 99*, pages 57–66, 1999.

[31] Y.-L. Wu, D. Agrawal, and A. E. Abbadi. A comparison of dft and dwt based similarity search in time-series databases. In *Proceedings of the 9 th International Conference on Information and Knowledge Management*, 2000.

[32] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 385–394. Morgan Kaufmann, 2000.

[33] B.-K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *ICDE*, pages 201–208, 1998.

[34] Y. Zhu, M. S. Kankanhalli, and C. Xu. Pitch tracking and melody slope matching for song retrieval. In *Advances in Multimedia Information Processing - PCM 2001, Second IEEE Pacific Rim Conference on Multimedia, Bejing, China, October 24-26*, 2001.

[35] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB 2002,Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*, pages 358–369, 2002.