Similarity-based Retrieval of Time-Series Data Using Multi-Scale Histograms

Lei Chen, M. Tamer Özsu
University of Waterloo
School of Computer Science
Waterloo, Canada
{l6chen,tozsu}@uwaterloo.ca

Technical Report CS-2003-31 Sept 2003

Abstract

Queries for similarity-based time series data retrieval can be categorized into two types: (1) pattern existence queries, and (2) exact match queries. Pattern existence queries find the time series data with certain patterns while exact match queries search over the time series data by specifying exact values and detailed temporal information. In this paper, we propose multi-scale time series histograms that can be used to answer both types of queries, thus offering users more flexibility. The experimental results show that multi-scale histograms can effectively find the patterns in time series data as well as answer exact match queries, even when the data contain noise, local time shifting, local time scaling, amplitude shifting and amplitude scaling.

1 Introduction

Similarity-based time series data retrieval has been attracting increasing interest in database and knowledge discovery communities because of its wide use in various applications, such as stock data or weather data analysis. Basically, two types of queries on time series data are studied: pattern existence queries [2, 25, 23], and exact match queries [1, 6, 8, 24, 32, 15, 11, 4].

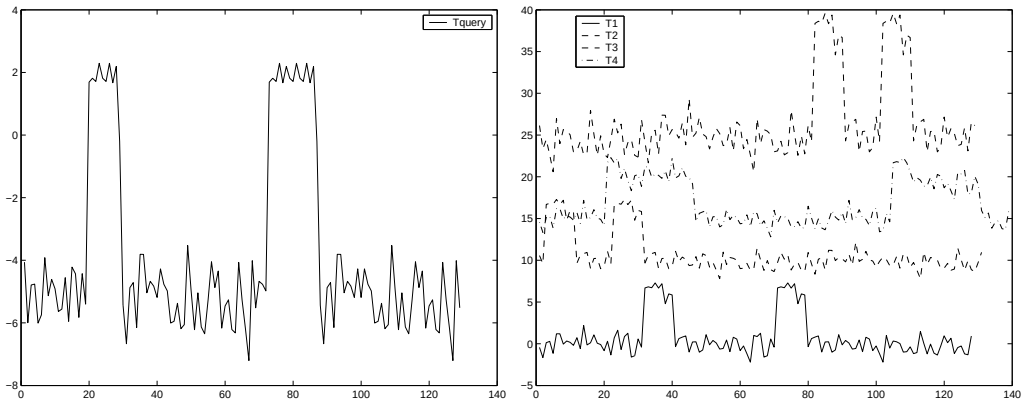
In pattern existence queries, users are interested in the general shape of time series data and ignore the specific details. For example,

- "Give me all the stock data of last month that have a *head and shoulder pattern*"; or
- "Give me all the temperature data of patients in last 24 hours that have *two peaks*".

The second query, for example, is used for detecting "goalpost fever", one of the symptoms of Hodgkin's disease, in the time series data that contain peaks exactly twice within 24 hours [25]. Figure 1(a) shows an example time series with *two peaks*. Using this as query data, a *two peaks* existence query should retrieve various time series that contain *two peaks* as shown in 1(b). Therefore, for pattern existence queries, as long as the time series data have the specified pattern, they will be retrieved, no matter when the pattern appears and how it appears.

The retrieval techniques for pattern existence queries should be invariant to the following:

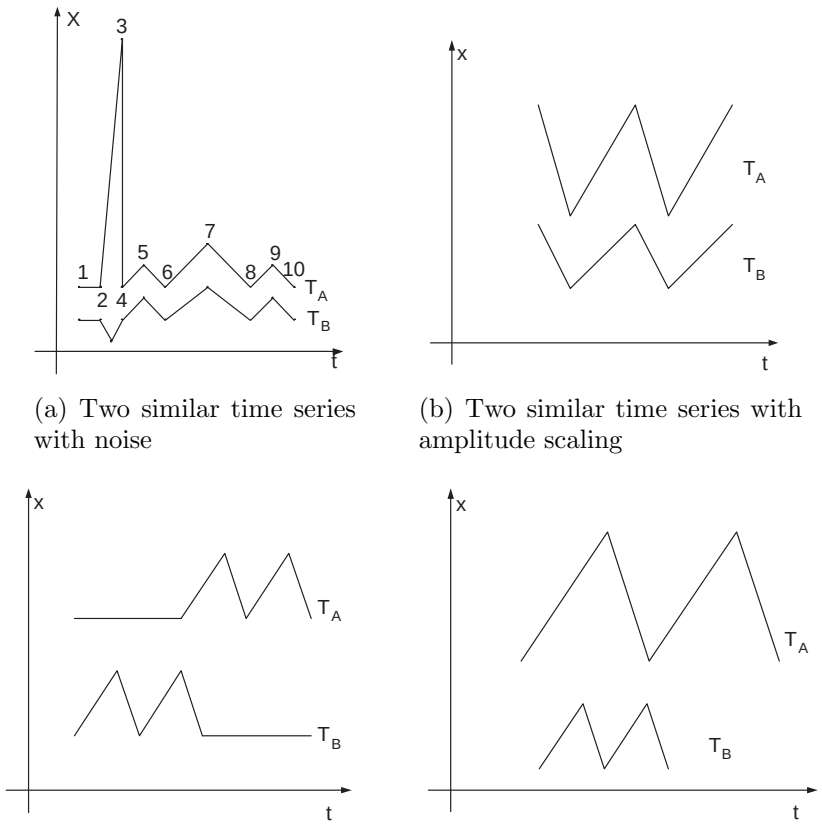
- **Noise.** In Figure 2(a), two similar time series T_A and T_B are shown. However, position 3 (which could be a noise data point) will introduce a large difference value when the Euclidean distance is computed between two time series, which may cause them to be determined as dissimilar.



(a) An example query time series with two peaks

(b) Various candidate time series contain two peaks

Figure 1: A pattern existence query on two peaks



(a) Two similar time series with noise

(b) Two similar time series with amplitude scaling

(c) Two similar time series with time shifting

(d) Two similar time series with time scaling

Figure 2: The different factors that may affect the similarity between two time series

- **Amplitude scaling and shifting.** In Figure 2(b), the two time series are similar in terms of the pattern that they contain (they both have "two bottom" pattern), but the amplitudes are different.

- **Time shifting and scaling.** In Figures 2(c) and 2(d), time series T_A and T_B will be considered dissimilar if we simply compare their positions. However, it can be argued that T_A and T_B are similar because they contain the same pattern ("two peaks").

Several approximation approaches [2, 25, 23] have been proposed to transform time series data to character strings over a discrete alphabet and apply string matching techniques to find the patterns. However, the transformation process is sensitive to noise. Furthermore, because of the quantization of the value space for transforming time series data into strings, data points located near the boundaries of two quantized subspaces may be assigned to different alphabets. As a consequence, they are considered to be different by string comparison techniques.

For exact match queries, the exact result of a query is defined in terms of specific values. The actual match results are the time series data that are within a specific threshold to the exact result. For example, "Give me all the stock data of last month that is similar to the IBM's stock data of last month". As shown in Figure 3, we are looking for the stock data within the boundaries that are described by two dashed curves. Most of the previous work focus on solving exact match queries. Euclidean distances are widely used to measure the similarity between two time series [1, 6, 24]. However, it requires two time series to have the same length and same sampling rate. Dynamic time warping (DTW) is proposed to measure two time series with different lengths and different sampling rates [3, 32, 15, 11]. However, DTW is sensitive to noise since it also requires matching all the data points, even for outliers [27]. Bozkaya et al. [4] present a modified version of edit distance, called longest common subsequences (LCSS), to compute the distance between two sequences of feature vectors, which were extracted from image sequences. However, a threshold must be set to determine whether or not the elements from two sequences match. It is difficult to find a suitable threshold value without prior knowledge of data. Therefore, a robust and easy-to-set similarity measure is needed for exact match queries.

To the best of our knowledge, there are no techniques that have been developed to answer both pattern existence queries and exact match queries. A typical application that needs answers to both queries is an interactive analysis of time series data. For example, users may be initially interested in retrieving all the time series data that have some specific patterns that can be quickly answered by pattern existence queries. After that, they may want to retrieve all the time series which are similar to an interesting time series that they find from the previous retrieval results. In this case, they can use

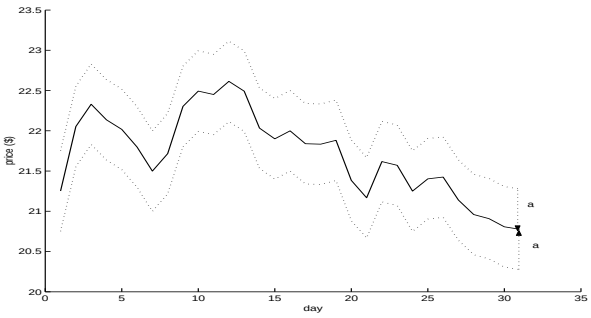


Figure 3: An exact match query using Euclidean distance

exact match queries.

In this paper, based on the observation that data distributions can capture patterns of time series data, we propose a multi-scale histogram-based representation to approximate time series data, which is invariant to noise, amplitude scaling and shifting, and time shifting. The multi-scale representation can answer both pattern existence queries and exact match queries and offers users flexibility to search time series data with different precision levels (scales). Weighted Euclidean distance is used to measure the similarity between two time series with the consideration of bin similarity. We also investigate two different approaches in constructing histograms: *equal bin size* and *equal area size*. The experimental results show that our histogram-based representation, together with the weighted distance measure, can effectively capture the patterns of time series data as well as the exact values of time series data.

The rest of the paper is arranged as follows: Section 2 presents concepts of time series histograms and two different types of time series histograms. We present multi-scale time series histograms in Section 3. In Section 4, we present experimental results using multi-scale time series histograms on finding patterns and answering exact match queries, followed, in Section 5, by a comparison of our work with earlier proposals. We conclude in Section 6 and indicate some further work.

2 Histogram-based data representation

A *time series data* T is defined as a sequence of pairs, each of which shows the value (x_i) that is sampled at a specific time denoted by a timestamp (t_i): $T = [(x_1, t_1), \ldots, (x_n, t_n)]$ where n, the number of data points in T, is defined as the *length* of T. We refer to this sequence as the *raw representation* of the time series data.

Given a set of N time series data $\mathcal{T}$, we first normalize each time series

T_i into its normal form using its mean (μ) and variance (σ) [8]:

$$Norm(T) = [(\frac{x_1 - \mu}{\sigma}, t_1), \ldots, (\frac{x_n - \mu}{\sigma}, t_n)]$$

The similarity measures computed from time series normal form are invariant to amplitude scaling and shifting.

Given the definition and normal form representation of a time series, we use time series histograms to compare them. Given the maximum and minimum values of normalized time series data as $max_{\mathcal{T}}$ and $min_{\mathcal{T}}$, respectively, we divide the range $[min_{\mathcal{T}}, max_{\mathcal{T}}]$ into τ disjoint equal size sub-regions, called *histogram bins*. Given a time sequence T, we can compute the histogram H of T by counting the number of data points h_i ($1 \leq i \leq n$) that are located in each histogram bin i: $H = [h_1, \ldots, h_\tau]$.

We normalize the time series histogram by dividing the value of each histogram bin i by the total number of data points in the time series. Since time series histograms are computed from normal form of time series data, the distance that is computed from two time series histograms are invariant to amplitude scaling and shifting. Furthermore, because time series histograms ignore the temporal information, they are also robust to time shifting and scaling. For example, in Figures 2(c) and 2(d), the histogram of normalized T_A is the same as that of T_B. Moreover, since time series histograms show the whole distribution of the data, and noise only make up a very small portion, comparisons based on histograms can remove the disturbance caused by noise. Therefore, time series histograms are ideal representations for answering pattern existence queries. Algorithm 1 describes how to construct time series histograms for data set $\mathcal{T}$.

L_1 or L_2 norms [26] can be used to measure the similarity between two histograms. However, these do not take the similarity between time series histogram bins into consideration, which may lead to poor comparison results. Consider three histograms H_1, H_2 and H_3 representing three time series of equal length. Assume that H_1 and H_2 have the same value on consecutive bins and H_3 has the same value in a bin which is quite far away from these bins of H_1 and H_2. L_1 and L_2 distances between any two of these three histograms are equal. However, for answering exact match queries, H_1 is closer to H_2 than it is to H_3. Even for pattern existence queries, data points which are located near the boundary of two histogram bins should be treated differently compared to those points that are far apart, which is also not considered by L_1 and L_2 distance measures.

A weighted Euclidean distance can be used to compute the distance between two color histograms [10]. We adapt this distance function to measure the similarity between two time series histograms. Given two time series T_A and T_B, the *weighted Euclidean distance* (WED) between their time series histograms H_A and H_B is:

Algorithm 1 The algorithm for constructing histograms for data set $\mathcal{T}$

Require: /*input: a time series data set $\mathcal{T}$ and the number of histogram bins τ */
Ensure: /*output: a histogram data set */

```
find max_T and min_T of the data set T
divide [min_T, max_T] into τ disjoint equal size histogram bins h_i
for all each time series T_i of T do
    for each data point x_i of T_i do
        for each histogram bin h_i do
            if h_{i,lowerbound} ≤ x_i < h_{i,upperbound} then
                h_i = h_i + 1 ;
                break;
            end if
        end for
    end for
    insert generated H_i to the result data set
end for
return the result data set
```

$$WED(H_A, H_B) = Z^T A Z$$

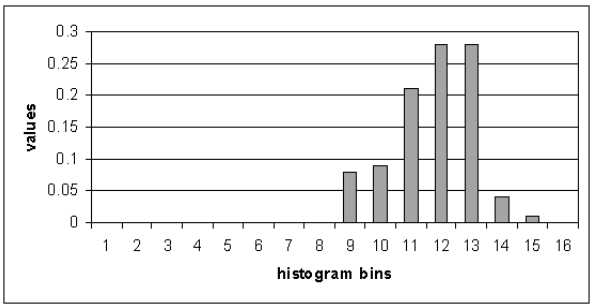
where $Z = (H_A - H_B)$, Z^T is the transpose of Z, and $A = [a_{ij}]$ is a similarity matrix whose element a_{ij} denotes similarity between two time series histogram bins i and j. The larger is a_{ij}, the more similar are bins i and j. We define a_{ij} as: $a_{ij} = (1 - |j - i|/\tau)$, where τ is the number of bins. The algorithm for answering pattern existence queries is given in Algorithm 2.

Algorithm 2 The algorithm for answering pattern existence queries

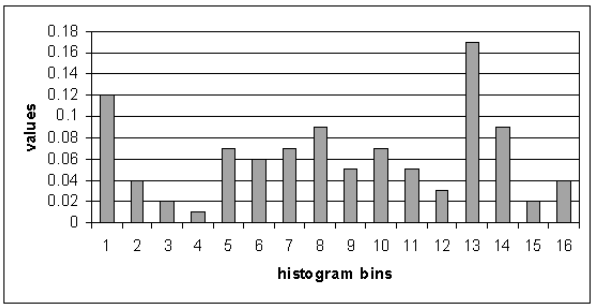
Require: /*input: A example time series T, its histogram H, a matching threshold ϵ. T contains the specified patterns*/
Ensure: /*output: A list of time series who contain the specified patterns*/

```
for all time series histograms H_i of T_i in the database do
    if WED(H_i, H) ≤ ε then
        insert the time series id i into the result list
    end if
end for
return the result list
```

In Algorithm 2, a matching threshold has to be set to determine whether the examined time series contains a pattern similar to that of the query time series. In our experiments, we find a suitable threshold based on the histogram of the query time series.



(a) Values of a 16 bin equal bin size histogram



(b) Values of a 16 bin equal area size histogram

Figure 4: A comparison of data distributions of an equal bin size histogram and an equal area size histogram

In the previous section, we defined a histogram with equal bin sizes. However, values of time series data normally are not uniformly distributed, leaving a lot of bins empty. In such cases, computing the distances between two time series histograms that contain many zeros is not helpful to differentiate the corresponding time series data. It has been claimed [19] that distributions of most time series data follow normal distribution. We have verified this phenomenon on the data sets that we use in our experiments.

Consequently, instead of segmenting the value space into τ equal size sub-regions, we segment the value space into sub-regions (called sub-spaces) that have the same area size under the normal distribution curve of that data. The boundary of each subspace can be computed as follows. Assuming that x_i is the lower bound of subspace i and x_{i+1} is the upper bound: $\int_{min_{\mathcal{T}}}^{max_{\mathcal{T}}} p(x)dx = \sum \int_{x_i}^{x_{i+1}} p(x)dx$ where $p(x)$ is the normal distribution function, $1 \leq i \leq \tau$, $x_1 = min_{\mathcal{T}}$, and $x_{\tau+1} = max_{\mathcal{T}}$. Even though we use normal distribution function to create equal area size histogram bins, the idea can be easily extended to other data distributions.

With equal area size segmentation, we assign equal probability to each histogram bin that data points of time series fall in. For example, for the "cameramouse" data that we used in our experiment, the average filling ratio of 16 bin equal area size histograms is about 98%. However, it is only 40% for the 16 bin equal bin size histograms. Figure 4 shows the values of equal area size and equal bin size histograms of a randomly selected data from "cameramouse" data set. From Figure 4, we can see that the values in the equal area size histogram are well distributed and values in the equal bin size histogram concentrate on a few bins.

In our experiments, we compare the effectiveness of equal bin size histograms and equal area size histograms in terms of classification accuracy. By changing line 2 of Algorithm 1 to construct histogram bins with equal area size, we can use the same algorithm to construct equal area size histograms.

3 Multi-scale Time Series Histograms

The time series histograms, as defined in the previous section, give a global view of the data distribution of time series data. However, they do not consider the temporal appearance order of values. For example, in Figure 5, two time series, T_A and T_B, are quite different in terms of appearance order. However, they have the same histograms as shown in Figures 6(a) and 6(c).

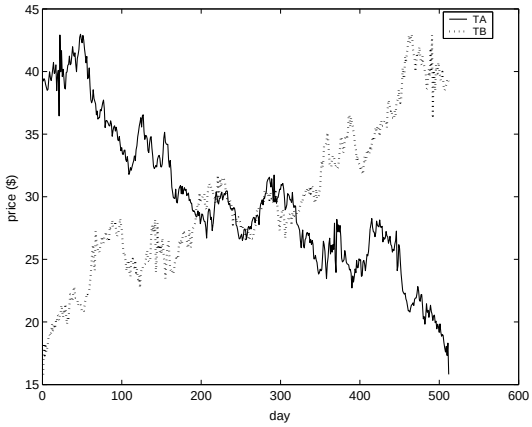


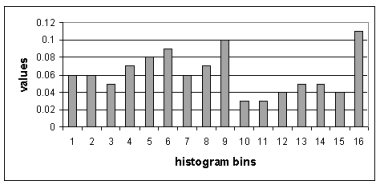
Figure 5: Two different time series with the same time series histogram

A multi-scale representation of time series histogram is designed for better discrimination of time series data based on their temporal details to facilitate exact match queries. A time series T_A of length n, can be equally divided into δ ($1 \leq \delta \leq n$) segments. For each segment, we can compute its time series histogram. If $\delta = 1$, we get the histogram for the entire time series as defined in the previous section. If further precision is required in comparing two time series, one can set $\delta > 1$, build a histogram for each segment (for both time series) and compare the corresponding histograms. Thus, we can develop multi-scale histograms on time series data by proper setting of δ and a specific δ value defines the precision level (i.e. scale) of the histogram. Figure 6 shows 2-scale histograms for time series T_A and T_B of Figure 5. In Figure 6, T_A and T_B can be easily distinguished at the second level ($\delta = 2$).

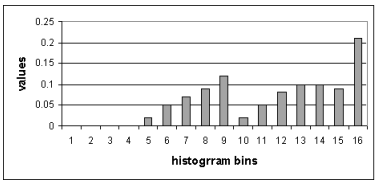
With multi-scale time series histograms, the exact match queries can be answered at several precision levels. Users can specify the scale levels when they submit a query. We use the average of the weighted Euclidean distances as the result of comparison at scale level δ:

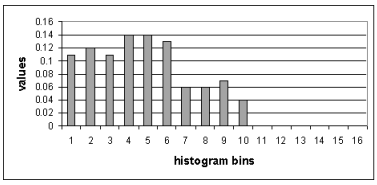
$$D_\delta = \frac{\sum_{i=1}^{\delta} WED(H_{A,i}, H_{B,i})}{\delta}$$

By extending the Theorem for color histograms in [20], we have:

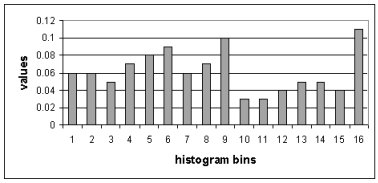


(a) a histogram of T_A at scale level 1

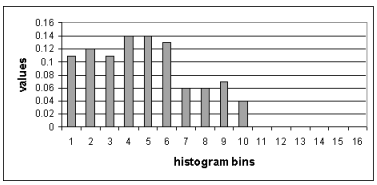


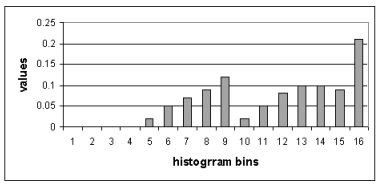


(b) two histograms of T_A at scale level 2



(c) a histogram of T_B at scale level 1





(d) two histograms of T_B at scale level 2

Figure 6: Two scale histograms of T_A and T_B

Theorem 1 In a n-level multi-scale time series histogram, if D_l denotes the distance between two time series histograms at scale level l, then $D_{l-1} \leq D_l \leq D_{l+1}$, where $2 \leq l \leq n$.

Proof: Straightforward extension from proof in [20].

When we answer an exact match query at higher scale level l, instead of directly computing the WED at level l,we can start computing the WED for each candidate time series at level 1, and then level 2 and so on [18]. According to Theorem 1, this multi-step filtering processing will not introduce false dismissals.

For both pattern existence queries and exact match queries, directly comparing τ-dimensional time series histograms is computationally expensive, even with the help of some multidimensional access methods (MAM) such as the R-tree [9]. Since τ is higher than 8-12 dimensions, the performance will be worse than sequential scan [30]. Therefore, we use the weighted average

of time series histogram to avoid comparisons on full histograms.

The average of a time series histogram is defined as the weighted average. Suppose $C = [c_1, c_2, \ldots, c_\tau]$ is a $1 \times \tau$ matrix whose ith column c_i is the upper bound value of histogram bin i (x_{i+1}). Given two τ dimensional time series histograms X and Y, the average histograms are: $X_{avg} = CX$ and $Y_{avg} = CY$. The squared average distance between X_{avg} and Y_{avg} is defined as: $WAD = (X_{avg} - Y_{avg})^T(X_{avg} - Y_{avg}) = (X - Y)^T C^T C(X - Y)$.

Theorem 2 [10] With WED and WAD defined as above, if $\tilde{A}$ is positive semidefinite, then for all vectors X and Y, $WAD \leq WED/\sqrt{\lambda_1}$, where λ_1 is the minimum eigenvalue of the generalized eigenvalue problem $\tilde{A}\tilde{Z} = \lambda\tilde{W}\tilde{Z}$ where $W = C^T C$, $\tilde{W}$ is the first $n - 1$ elements of W, $Z = (X - Y)$, $\tilde{Z}$ and $\tilde{A}$ are defined with respect to Z and A, respectively.

Therefore, instead of directly computing the weighted histograms, we can first compute the distance between the weighted averages of two time series histograms to prune false alarms from the database. Weighted averages can be considered as the first filter when we use multi-step filtering to answer an exact match query at a higher scale l. The algorithm for multi-step filtering is given in Algorithm 3.

4 Experiments and Discussion

In this section, we present the results of some experiments that we conducted to evaluate the efficacy and robustness of the histogram-based similarity measure and the matching algorithm. All the programs are written in C and run on a Sun-Blade-1000 workstation under Solaris 2.8 with 1GB of memory.

4.1 Efficacy and Robustness of Similarity Measures

Our first experiment is designed to test how well the weighted Euclidean distances computed from histograms perform in finding patterns in time series data. We first compare our approach with the approach proposed in [25] on the labelled time series data sets: Cylinder-Bell-Funnel (CBF). The reason for selecting CBF data sets is that it contains very simple distinct patterns, allowing a direct comparison of the techniques. The CBF data set has three distinct classes of times series: cylinder, bell and funnel as shown in Figure 7. We generate 1000 CBF data sets with 100 examples for each class. We implemented Shatkay's algorithms using the best line fitting. In order to capture the overall shapes, a larger error threshold (1.2) was used. However the general shape of bell and funnel are treated as similar in [25], because both of them contain a *peak*. Therefore, for fairness, we only used cylinder and bell data sets to test for existence queries. In order to test robustness of the similarity measures, we added random Gaussian noise and local time

warping to both data sets using a modified program of [28]. We added to this program non-interpolated noise and large time warping ($20 - 30\%$ of the series length) since pattern existence queries are only interested in the existence of a given pattern. Figure 8 shows the best fitting line segments (dashed lines) for the cylinder and bell data sets that contain noise and time wrapping. The slope of each line segment is mapped into a symbol according to the predefined mapping tables (e.g. Table 1 [23]) and consecutive symbols are connected together to form a string.

Algorithm 3 The algorithm for answering exact match queries with multi-scale filtering

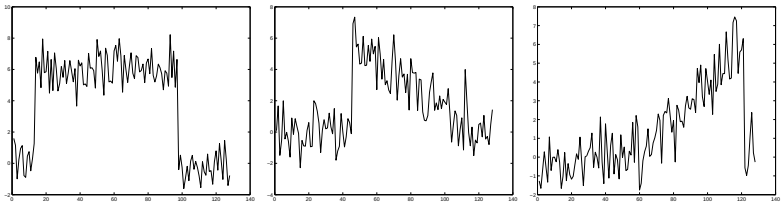
Require: /*input: A example time series T, its histogram H, a matching threshold ϵ. precision level δ*/
Ensure: /*output: A list of time series who match T*/

```
for all multi-scale time series histograms H_i of T_i in the database do
    Compute WAD between the weighted average of H and that of H_i
    if WAD ≤ ε/√λ_1 then
        insert the time series id i into the result list R_0
    end if
end for
j = 1
repeat
    for each i in R_{j-1} do
        if WED(H_i, H) ≤ ε then
            insert the time series id i into the result list R_j
        end if
    end for
    j = j + 1
until (j == δ + 1) or (R_{j-1} is empty)
if R_{j-1} is empty then
    return NULL
else
    return the result list R_δ
end if
```

Slope symbol	Slope partiton
U(Up)	$(tan(\pi/6), \infty)$
F(flat)	$(tan(-\pi/6), tan(\pi/6))$
D(Down)	$(-\infty, tan(-\pi/6))$

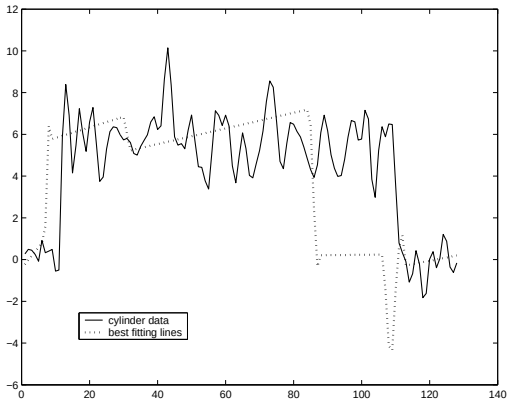
Table 1: An example of mapping quantized slope subspaces into symbols

In [25], regular expressions are used to query the converted strings for

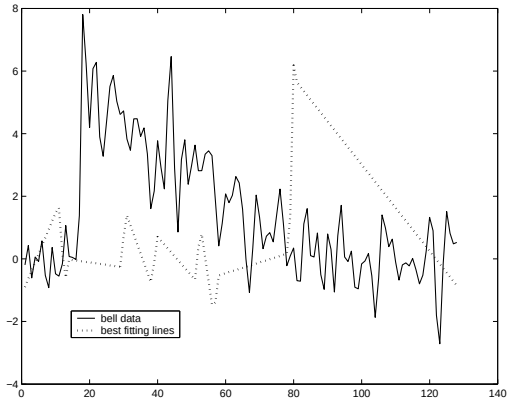


(a) Cylinder (b) Bell (c) Funnel

Figure 7: Cylinder-Bell-Funnel data set



(a) best fitting lines for cylinder time series



(b) best fitting lines for bell time series

Figure 8: Best fittings lines for cylinder and bell data

certain patterns. The regular expression for retrieving time series that contain cylinder shape is $F^*UF^+DF^*$ and F^*UDF^* for bell shape. We use Algorithm 1 to search the patterns in the time series data. We generated another "pure" CBF data set of size 150 as query data, each class contains 50 examples. The histograms are also computed from the query data set. We use precision and recall to measure our retrieval results which are well known metrics in information retrieval. Precision measures the proportion of correctly found time series, while recall measures the proportion of cor-

rect time series that are detected. In this experiment, we need to determine the matching threshold ϵ. We tested several values and found that half the weighted Euclidean distance between the query histogram and an empty histogram gave the best results. We run the query for 50 times and averaged the results, as shown in Table 2. Even though the regular expression approach achieves relatively high precision, its recall is very low. This is the effect of noise. Our histogram-based approach can achieve relatively high precision and recall, which confirms that our approach is suitable for answering pattern existence queries. From Table 2, we also find that dividing the value space into too many sub-regions (histogram bins) does not improve the results significantly; we can achieve reasonably good results around 16 bins.

	Cylinder		Bell	
	precision	recall	precision	recall
regular expression	81.48	44.00	75.61	31.00
his($\tau = 8$)	72.22	91.00	63.39	71.00
his($\tau = 16$)	72.44	92.00	78.43	80.00
his($\tau = 32$)	75.63	90.00	79.28	88.00
his($\tau = 64$)	72.58	90.00	79.44	85.00

Table 2: Precision and Recall on Pattern Existences Queries

The second experiment is designed to check the effectiveness and robustness of multi-scale histograms in answering exact match queries. Since the objective measurements of the quality of a proposed similarity measure can be readily obtained by running simple classification experiments on labelled data sets [14], we first compare the classification error rates in results produced by the weighted Euclidean distances to those of DTW and LCSS by evaluating them on the Control-Chart (CC) data set. We compare WED with DTW and LCSS because DTW and LCSS can also handle local time shifting or noise disturbances. The classification error rate is defined as a ratio of the number of misclassified time series to the total number of the time series. There are six different classes of control charts: "normal", "cyclic", "increasing trend", "decreasing trend", "upward shift", and "downward shift". Each class contains 100 examples. We later tested our techniques using more complicated data sets. We added non-interpolated noise and time warping ($10 - 20\%$ of the series length) to test the robustness of weighted Euclidean distance in answering exact match queries. For simplicity, we carried out a simple classification using 1-Nearest Neighbor with three similarity measures and checked the classification results using the "leave one out" verification mechanism.[1] We repeat the experiment on all the time series of CC data

[1]The "leave one out" verification mechanism takes one sample data from a class and finds a nearest neighbor of the data in the entire data set by applying the predefined

set. The error rates using DTW and LCSS are 0.127 and 0.163, respectively. It has been claimed that LCSS is more accurate than DTW [27]. However, our experiments could not replicate this result. The possible reason for this discrepancy is the difference in the choice of the matching threshold value. This experiment also verifies the claim that we made in Section 1, namely that it is really difficult to set a proper matching threshold for LCSS.

We report comparable results using the weighted Euclidean distances in Table 3. We run the experiment with different number of bins (τ) and scales (δ) using WED on time series histograms with equal bin size.

	number of bins τ			
scale δ	8	16	32	64
1	0.628	0.590	0.563	0.536
2	0.223	0.157	0.123	0.130
3	0.257	0.158	0.140	0.127
4	0.223	0.158	0.140	0.130
5	0.167	0.143	0.128	0.123
6	**0.140**	**0.105**	**0.105**	**0.100**
7	0.172	0.137	0.126	0.268
8	0.182	0.137	0.108	0.298
9	0.190	0.137	0.117	0.253
10	0.122	0.137	0.105	0.248
11	0.168	0.130	0.118	0.298
12	0.153	0.148	0.125	0.298
13	0.197	0.150	0.137	0.268
14	0.243	0.168	0.143	0.298
15	0.243	0.188	0.143	0.298
16	0.243	0.188	0.143	0.298

Table 3: Error rates using WED on time series histograms with equal bin size

Comparing the error rates of DTW (0.127), LCSS (0.163) and those of Table 3, we observe that WED performs better than the other two similarity measures in answering exact match queries. From Table 3, we find an interesting fact that very high scales (i,e., high values of δ) may lead to worse classification results. This is because the higher the scale, the more temporal details will be involved in computing WED, which causes time series histograms at that scale level to be more sensitive to local time shifting and scaling. Another fact demonstrated in Table 3 is that higher number of bins may not lead to more accurate classification results. This is because the higher the number of bins, the more detailed information will be captured

distance metrics. If the found nearest neighbor belongs to the same class as the sample data, it is a hit, otherwise, it is a miss.

in time series histogram, including noise. These characteristics of multi-scale time series histograms exactly fit our needs, since they suggest that we only need to check the first few scale histograms with small number of histogram bins. However, compared to results of DTW or LCSS, the improvements reported in Table 3 are modest, especially for the first few scale histograms (i.e. lower values of δ).

To better understand these results, we investigated the filling ratio of equal bin size histograms. The filling ratio is defined as the number of non-empty bins to the total number of bins. We found that nearly 50% of the bins are empty! Consequently, WED between two time series histograms are not able to distinguish them properly, since most of the bins are identical! Therefore, we run the same experiment on time series histograms with equal area size. Table 4 reports the results.

	number of bins τ			
scale δ	8	16	32	64
1	0.168	0.116	0.123	0.112
2	0.098	0.078	0.783	0.069
3	0.085	0.070	0.058	0.063
4	**0.073**	**0.050**	**0.068**	**0.065**
5	0.073	0.055	0.068	0.052
6	0.063	0.058	0.055	0.095
7	0.107	0.083	0.090	0.135
8	0.102	0.090	0.086	0.106
9	0.145	0.120	0.121	0.132
10	0.132	0.107	0.115	0.127
11	0.143	0.123	0.133	0.157
12	0.153	0.128	0.137	0.153
13	0.167	0.155	0.167	0.151
14	0.147	0.131	0.151	0.167
15	0.167	0.158	0.163	0.163
16	0.167	0.158	0.167	0.169

Table 4: Error rates using WED on time series histograms with equal area size

The results of Tables 3 and 4 demonstrate that the improvement when equal area size is used is nearly 2 times for CC data! The filling ratio of time series histogram is around 80%. Table 4 also shows that using only the first few scales (e.g. $\delta = 4$), provides reasonably high accuracy.

In our third experiment, we test the matching accuracy of multi-scale time histograms versus DTW and LCSS on classifying time series data with more complicated shapes, such as the time series data in Figure 9. Classifying these types of time series requires matching on temporal details of the data.

We evaluate WED, DTW and LCSS by two labelled trajectory data sets that are generated from the Australian Sign Language (ASL)[2] data and the "cameramouse" [7] data. Only x positions are used for both data sets. We first select a "seed" time series from each of the two data sets and then create two additional data sets by adding interpolated Gaussian noise and small time warping (5-10% of the time series length) to these seeds [28]. The ASL data set from UCI data consists of samples of Australian Sign Language signs. Various parameters were recorded when a signer was signing one of 95 words in ASL. We extracted one recording from each of following 9 words as "seed" time series of ASL data: "Norway", "cold", "crazy", "eat", "forget", "happy", "innocent", "later", and "lose" as shown in Figure 9. The "cameramouse" data set contains 15 trajectories of 5 words (3 for each word) obtained by tracking the finger tip when people write various words. We use all trajectories of each word as the seeds. The data set that is generated from seeds of "cameramouse" contains 5 classes and 30 examples of each class, while the one from ASL seeds contains 9 classes and each class has 10 examples. We use the same classification and verification algorithms as in the second experiment. Based on the observation of the second experiment, we use time series histogram with same area size. Tables 5 and 6 report the results.

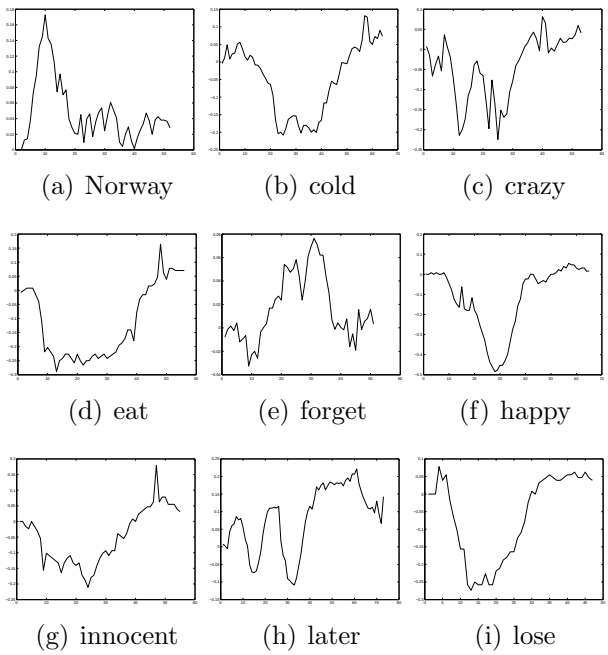


Figure 9: ASL data set

The results of Tables 5 and 6 show that WED again achieves better classification result. For both data sets, WED can get 0 error rate. A surprising

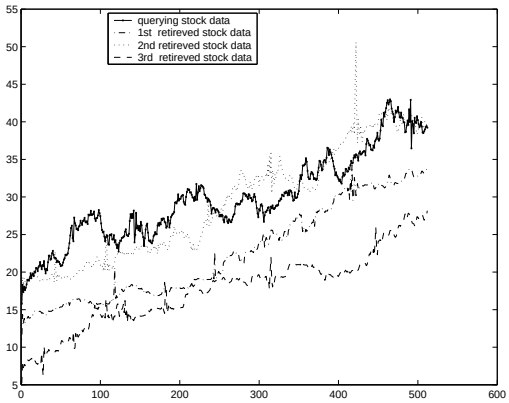
[2]http://kdd.ics.uci.edu

	ASL data	Cameramouse data
DTW	0.09	0.08
LCSS	0.29	0.30

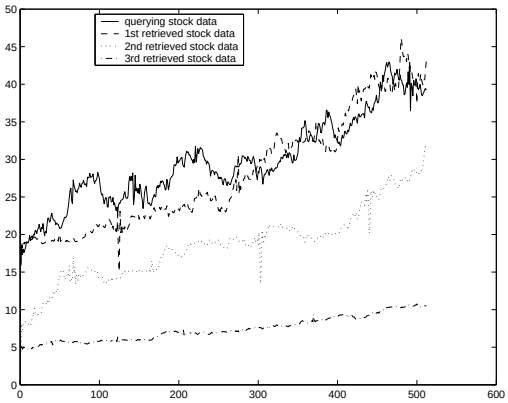
Table 5: Error rates of LCSS and DTW

	ASL DATA				Cameramouse DATA			
	number of bins τ				number of bins τ			
scale	8	16	32	64	8	16	32	64
1	0.07778	0.05555	0.04444	0.03333	0.13333	0.01333	0.06667	0.07333
2	**0.05556**	**0.04444**	**0.02222**	**0.02222**	**0.01333**	**0.01333**	**0.03333**	**0.04**
3	0.05556	0.03333	0.02222	0.04444	0.02	0.06	0.01333	0.01333
4	0.05556	0.05556	0.04444	0.04444	0.02667	0.0.2	0.01333	0.01333
5	0.03333	0.04444	0.03333	0.04444	0.08667	0.02	0.01333	0.03333
6	0.04444	0.05556	0.02222	0.03333	0.04	0.01333	0	0.01333
7	0.05556	0.02222	0.03333	0	0.04	0.01333	0	0.01333
8	0.03333	0.04444	0.04444	0	0.04667	0.02	0.04	0.04667
9	0.03333	0.05556	0.05556	0.04444	0.04667	0.01333	0.01333	0.06
10	0.02222	0.05556	0.02222	0.02222	0.02	0.06667	0.04	0.04
11	0.02222	0.04444	0.02222	0.02222	0.04667	0.05333	0.03333	0.03333
12	0.05556	0.02222	0.02222	0.03333	0.02	0.03333	0.04	0.05333
13	0.02222	0.05556	0.02222	0.02222	0.05333	0.05333	0.02667	0.04
14	0.05556	0.02222	0.03333	0.02222	0.01333	0.04	0.01333	0.05333
15	0.03333	0.04444	0.03333	0.02222	0.04	0.04667	0.05333	0.05333
16	0.05556	0.02222	0.04444	0.02222	0.04	0.04	0.02	0.04667

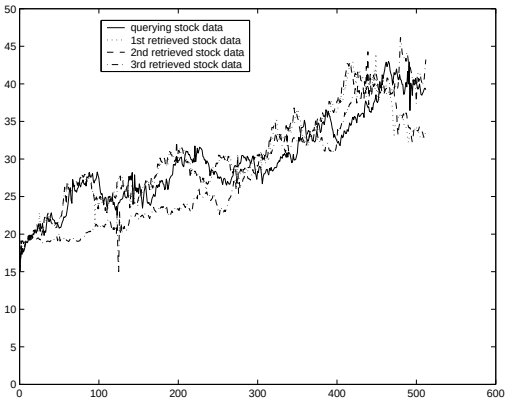
Table 6: Error rates using WED on time series histograms with equal area size



(a) 3-NN results on stock data using DTW



(b) 3-NN results on stock data using LCSS



(c) 3-NN results on stock data using WED

Figure 10: A comparison of 3-NN queries using DTW, LCSS and WED

observation is that at very low scale (e.g. $\delta = 2$), using WED can already achieve better results than DTW and LCSS. Through the investigation of the filling ratios of both histograms, we find that the number of empty bins only accounts for less than 5% of the total number of bins. The standard deviations of both data sets are also very high with respect to their mean values, which indicates that data points are widely spread in their value space. Compared with results of the second experiment, we conclude that WED on time series data whose data points are widely distributed in their value space (higher histogram filling ratio) can achieve better classification accuracy even at lower scales.

In our fourth experiment, we test the matching quality of WED on multi-scale time histograms in answering k-nearest neighbor (k-NN) search. For comparison, we also run the experiment using DTW and LCSS. Given a time series T_A, k-NN search will return the top k time series in the database that are most similar to T_A. We use a real stock data set that contains 193 company stocks' daily closing price from late 1993 to early 1996, each consisting of 513 values [29]. We use each time series as a "seed" and create a new data set by adding interpolated Gaussian noise and small time warping (2-5% of the time series length) to each seed. The new data set contains 1930 time series (each seed is used to create 10 new time series). We randomly select 10% of the sequences as query data. The results are that WED always performs better DTW and LCSS. Figure 10 shows one of the results of 3-NN search using WED ($\delta = 6$, $\tau = 32$), DTW and LCSS on stock data.

4.2 Efficiency of Multi-Step Filtering

Theorem 1 in Section 3 showed that multi-step filtering using multi-scale time series histograms will not introduce false alarms. However, how many histogram comparisons are saved using multi-step filtering? We run the experiment on the real stock data set that is used in the fourth experiment. We randomly select a time series and conduct a range query at precision level 6. We compute the number of comparisons that are needed for searching using only level 6, using level 1 and 6, and using all 6 levels on different range thresholds. We run the experiments 100 times and the average results are reported in Table 7.

From Table 7, we can see that the step-by-step filtering is the best strategy to reduce the total number of comparisons for small thresholds. For large thresholds, directly computing the distance at a higher level is a better choice in terms of the number of comparisons. However, large thresholds are not very useful for finding the desirable results, since a larger portion of the data in the database will be returned.

The number of comparisons needed for computing the histogram distance only relates to the CPU computation cost. However, the I/O cost for

	level 6 only	level 1 and level 6	all 6 levels
$\epsilon = 0.5$	11580	13498	17190
$\epsilon = 0.2$	11580	12760	7590
$\epsilon = 0.1$	11580	6838	3668
$\epsilon = 0.05$	11580	2356	2072

Table 7: Number of comparisons of different filtering approaches

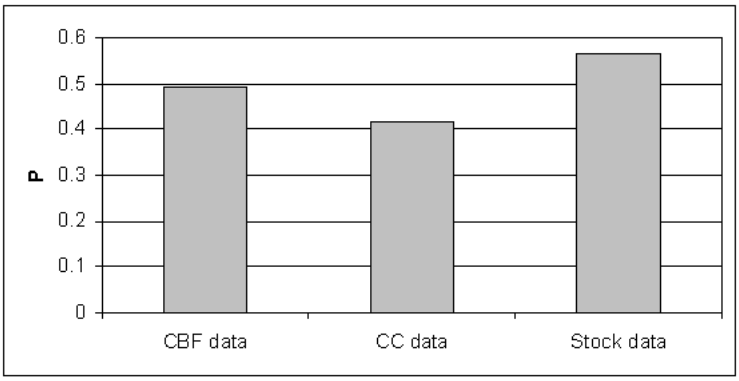


Figure 11: The punning power of lower bounding function

sequentially reading in the data files becomes a bottleneck with the growth of database size. If we can filter out the false alarms without reading in the data files, a significant speed-up will be achieved. For our time series histograms, we stored their average weighted histograms separately from histograms. The distances between the average histogram of query data and those of stored data are first computed to remove the possible false alarms. Therefore, the pruning power of average histograms is quite important. The pruning power (P), which is defined as the fraction of the database that must be examined before we can guarantee that we have found the nearest match to 1-Nearest Neighbor query [13]. Figure 11 shows the pruning power on different data sets of time series histograms with 16 bins (based on the previous experimental results, WED can already achieve reasonable good results when the histogram bin size is 16). Because "cameramouse" and ASL data sets are small, we did not include them in this experiment. Figure 11 demonstrates that using the average weighted Euclidean distance, we can remove nearly 40% of the false alarms, which will be helpful when the database size becomes large.

5 Related Work

As mentioned in Section 1, significant research has been done on similarity-based retrieval of time series data. The earliest proposal was by Agrawal et. al. [1], which used Euclidean distance to measure the similarity between time series data and applied Discrete Fourier Transform (DFT) to reduce the dimensionality for fast retrieval. Later, this work was extended to allow subsequence matching in [6]. Subsequent work have focused on two main aspects: (1) new dimensionality reduction techniques (assuming that the Euclidean distance is the similarity measure), such as Single Value Decomposition (SVD) [16, 12], Discrete Wavelet Transform (DWT) [17, 22], Piecewise Aggregate Approximation (PAA) [12, 31], and Adaptive Piecewise Constant Approximation (APAC) [11]; and (2) new similarity measures, other than the Euclidean distance, for measuring the similarity between two time series data. The motivation of the second research direction is that the Euclidean distance is too rigid and it is sensitive to noise, local time shifting, and scaling. Therefore, Berndt and Clifford [3] introduced Dynamic Time Warping (DTW) to measure the similarity between two time series. Compared to Euclidean distance, DTW allows time series data to "stretch" when they are compared to each other. However, directly computing DTW between two time series data is computational expensive, and several approaches [32, 15, 11] have been proposed to index on the lower bounds of DTW. Bozkaya et al. [4] used Longest Common Subsequences (LCSS) to measure the similarity between two image sequences, which was then applied to time series data [5, 27]. Perng et al. [21] proposed a Landmark model based on the human intuition and episodic memory. However, the Landmark model did not consider the effects of noise. All these techniques can be put into the category of exact match retrieval. For pattern existence retrieval, users are more interested in the existence of pattern in the time series data. A few approaches [2, 25, 23] have been proposed for finding movement patterns in time series data. The moving direction (the slope between two values) of an user specified interval [23], consecutive values [2], or a segment (obtained by a segmentation algorithm) [25] was represented as a distinct alphabet. Thus, their approaches converted the time series data into strings and could apply string-matching techniques to find the patterns.

Compared to all the previous work, our multi-scale time histogram-based similarity retrieval has the following advantages:

- Multi-scale time series histograms can be used to answer exact match and pattern existence queries.
- The weighted Euclidean distance takes the bin similarity into consideration, which reduces the boundary effect that introduced by value space quantization.

- Multi-scale time histograms are invariant to local time shifting and scaling, local amplitude shifting and scaling. Moreover, they can remove the disturbance introduced by noise.

- Multi-scale time histograms offer users flexibility to query the time series data on different accuracy level. Furthermore, lower scale histograms can be used to prune the false alarms from the database before we querying at higher scale histograms.

6 Conclusion and Future Work

In this paper, we propose a novel representation of time series data using multi-scale time series histograms. This representation is based on the intuition that the distribution of time series data can be an important cue for comparing similarity between time series. Multi-scale time series histograms are capable of answering both pattern existence queries and exact match queries. Moreover, they are invariant to local time shifting and scaling, and local amplitude shifting and scaling. Weighted Euclidean distance is used to measure the similarity between two time series histograms taking into account data points located near the boundaries of histogram bins. We also investigate two different types of histograms: equal bin size and equal area size.

Our experiments indicate that multi-scale time series histograms outperform string representations in finding patterns and outperform DTW and LCSS in answering exact match queries when the time series data contain noise or time shifting and scaling. The experiments also shows that equal area size histogram is more suitable for time series data comparison and weighted average histogram distance can effectively prune the false alarms from the database before computing the distance between two histograms.

In this paper, we only address the whole sequence matching problem. In future work, we plan to extend multi-scale histograms to subsequence matching. We also want to investigate the possibility of automatically setting up the scale value for users and to create some hierarchy indexing structures for fast access multi-scale time series histograms.

Acknowledgements: Thanks to Eamonn Keogh, Michalis Vlachos, Changzhou Wang, and X. Sean Wang for providing their source codes or data sets. This research is funded by Intelligent Robotics and Information Systems (IRIS), a Network of Center of Excellence of the Government of Canada.

References

[1] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *Proc. 4th Int. Conf. of Foundations of Data Organization and Algorithms*, pages 69–84, 1993.

[2] R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zaït. Querying shapes of histories. In *Proc. 21th Int. Conf. on Very Large Data Bases*, pages 502–514, 1995.

[3] D. J. Berndt and J. Clifford. Finding patterns in time series: A dynamic programming approach. In *Advances in Knowledge Discovery and Data Mining*, pages 229–248. AAA/MIT, 1996.

[4] T. Bozkaya, N. Yazdani, and Z. M. Ozsoyoglu. Matching and indexing sequences of different lengths. In *Proc. 6th Int. Conf. on Information and Knowledge Management*, pages 128–135, 1997.

[5] G. Das, D. Guopulos, and H. Mannila. Finding similar time series. In *Proc. 1st European Symp. on Principles of Data Mining and Knowledge Discovery*, pages 88–100, 1997.

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

[7] J. Gips, M. Betke, and P. Fleming. The camera mouse: Preliminary invertigation of automated visaul tracking for computer access. In *In Proc. Conf. on Rehabilitation Engineering and Assistive Technology Society of North America*, pages 98–100, 2000.

[8] D.Q. Goldin and P.C. Kanellakis. On similarity queries for time series data: Constraint specification and implementation. In *In Proc. of the Int. Conf. on Principles and Pratice of Constraint Programming*, pages 23–35, 1995.

[9] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 47–57, 1984.

[10] J. Hafner, H.S. Sawhney, W. Equitz, M. Flickner, and W. Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(7):729–739, 1995.

[11] E. Keogh. Exact indexing of dynamic time warping. In *Proc. 28th Int. Conf. on Very Large Data Bases*, pages 406–417, 2002.

[12] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2000.

[13] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 151–162, 2001.

[14] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. In *Proc. 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 102–111, 2002.

[15] S Kim, S. Park, and W. Chu. An indexed-based approach for similarity search supporting time warping in large sequence databases. In *Proc. 17th Int. Conf. on Data Engineering*, pages 607–614, 2001.

[16] F. Korn, H. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 289–300, 1997.

[17] K.P.Chan and A.W-C Fu. Efficient time series matching by wavelets. In *Proc. 15th Int. Conf. on Data Engineering*, pages 126–133, 1999.

[18] K. Leung and R. T. Ng. Multistage similarity matching for sub-image queries of arbitrary size. In *Proc. of 4th Working Conf. on Visual Database Systems*, pages 243–264, 1998.

[19] J. Lin, E. Keogh, S. Lonardi, and P. Patel. Finding motifs in time series. In *Proc. 2nd Int. Workshop Temporal Data Mining*, pages 370–377, 2002.

[20] S. Lin, M. T. Özsu, V. Oria, and R. Ng. Multi-precision similarity querying of image databases. In *Proc. 27th Int. Conf. on Very Large Data Bases*, pages 221–230, 2001.

[21] C.-S. Perng, H. Wang, S.R. Zhang, and D.S. Parker. Landmarks: a new model for similarity-based pattern querying in time series databases. In *Proc. 16th Int. Conf. on Data Engineering*, pages 33–42, 2000.

[22] I. Popivanov and R. J. Miller. Similarity search over time series data using wavelets. In *Proc. 17th Int. Conf. on Data Engineering*, pages 212–221, 2001.

[23] Y. Qu, C. Wang, and X. S. Wang. Supporting fast search in time series for movement patterns in multiple scales. In *Proc. 7th Int. Conf. on Information and Knowledge Management*, pages 251–258, 1998.

[24] D. Rafiei and A. O. Mendelzon. Efficient retrieval of similar time sequences using DFT. In *Proc. 9th Int. Conf. of Foundations of Data Organization and Algorithms*, 1998.

[25] H. Shatkay and S.B. Zdonik. Approximate queries and representations for large data sequences. In *Proc. 12th Int. Conf. on Data Engineering*, pages 536–545, 1996.

[26] M. J. Swain and D. H. Ballard. Color indexing. *Int. Journal of Computer Vision*, 7(1):11–32, 1991.

[27] M. Vlachos, G. Kollios, and D. Gunpulos. Discovering similar multidimensional trajectories. In *Proc. 18th Int. Conf. on Data Engineering*, pages 673 – 684, 2002.

[28] M. Vlachos, J. Lin, E. Keogh, and D. Gunopulos. A wavelet-based anytime algorithm for k-means clustering of time series. In *Proc. Workshop on Clustering High Dimensionality Data and Its Applications*, 2003.

[29] C.Z. Wang and X. Wang. Supporting content-based searches on time series via approximation. In *Proc. 12th Int. Conf. on Scientific and Statistical Database Management*, pages 69–81, 2000.

[30] R. Weber, H.-J Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proc. 24th Int. Conf. on Very Large Data Bases*, pages 194–205, 1998.

[31] B-K Yi and C. Faloutsos. Fast time sequence indexing for arbitrary Lp norms. In *Proc. 26th Int. Conf. on Very Large Data Bases*, pages 385–394, 2000.

[32] B-K Yi, H. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proc. 14th Int. Conf. on Data Engineering*, pages 23–27, 1998.