

# Literature Review on Time Series Indexing

Pavel V. Senin

Collaborative Software Development Lab

Department of Information and Computer Sciences

University of Hawaii

Honolulu, HI

senin@hawaii.edu

CSDL Technical Report 09-08

<http://csdl.ics.hawaii.edu/techreports/09-08/09-08.pdf>

April 2009

Copyright © Pavel V. Senin 2009

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Preamble . . . . .	1
1.2	Introduction to time-series . . . . .	1
<b>2</b>	<b>Time-series similarity measurements</b>	<b>5</b>
2.1	Euclidean distance . . . . .	7
2.2	Time series normalization . . . . .	8
2.2.1	Normalization into an Interval . . . . .	8
2.2.2	Normalization to Sum 1 and Normalization to Euclidean Norm 1 . . . . .	8
2.2.3	Normalization to the zero mean . . . . .	9
2.2.4	Normalization to Zero Mean and Unit Standard Deviation . . . . .	9
2.3	Transformation rules (smoothing, scales and shifts) . . . . .	10
2.4	Dynamic Time Warping (DTW) . . . . .	10
2.5	Longest Common Subsequence (LCS) . . . . .	12
<b>3</b>	<b>Dimensionality reduction, indexing and bounding</b>	<b>15</b>
3.1	Lower Bounding . . . . .	16
3.2	Discrete Fourier Transform (DFT) based decomposition . . . . .	16
3.3	Singular Value Decomposition (SVD) . . . . .	18
3.4	Discrete Wavelet Transform (DWT) decomposition . . . . .	19
3.5	Piecewise Approximation of time-series (PLA, PCA, PAA, APCA) . . . . .	21
3.6	Symbolic Aggregate approXimation (SAX) . . . . .	23
<b>4</b>	<b>Conclusion</b>	<b>26</b>

## **Abstract**

Similarity search in time-series databases has become an active research area in the past decade due to the tremendous growth of the amount of temporal data collected and publicly available. The complexity of this similarity problem lies in the high dimensionality of the temporal data making convenient methods inappropriate. The most promising approaches involve dimensionality reduction and indexing techniques which are the subject of this review. After starting with a general introduction to the time-series and classical time-series analyses, we will discuss in detail time-series normalization techniques and relevant distance metrics. We conclude with a review of the dimensionality-reduction and indexing methods proposed to date.

# Chapter 1

## Introduction

### 1.1 Preamble

This literature review provides background for the data mining toolkit I am developing to extend the open-source framework for automated collection of the software development process data named Hackystat. Hackystat gathers data about human activity (developers' behavior) and various code metrics indicating size, structure and quality of the code. After transformation and aggregation of the raw sensor data Hackystat stores it in a temporal database providing users with infrastructure for the retrieval and visualization of the telemetry data. It is possible not only to select a desired data granularity but also to refine telemetry streams by specifying individual or groups of developers, set of projects as well as individual metrics and their derivatives. While all of this provides a rich environment for the analysis and interpretation of the software development process, the comparison of telemetry streams is still performed by “eyeballing” and thus automation is a highly desirable feature. Once processing of range queries is implemented, it will allow extension of the analysis toolkit with various KDD methods aiming but not limited to indexing, clustering<sup>1</sup>, software development pattern discovery and real time sensor data stream analysis.

### 1.2 Introduction to time-series

The time-series is a data type represented by a sequence of data points sampled at successive times and usually found as the sequence of real or integer numbers with or without attached timestamps. Time-series data arises naturally from observations and reflects an evolution of some subject or a development of some phenomena in time. Since the time-series is the only way to store valuable and often non-reproducible temporal information, it makes time-series data ubiquitous and important not only in every scientific field but also in everyday life. For example, it is very common to see visualized time-series representing financial information about stocks and currency fluctuations,

---

<sup>1</sup>We are aware of controversial views on the clustering of time-series expressed in [39] by Keogh et al.

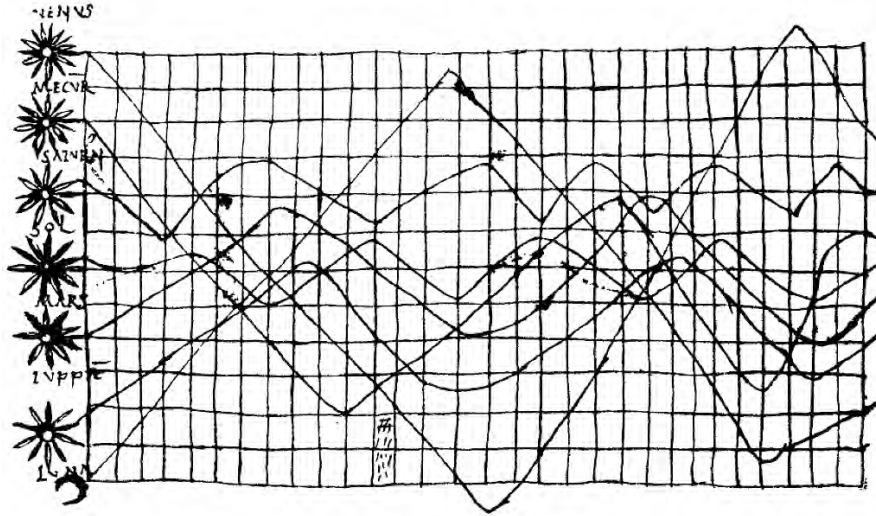


Figure 1.1: Ancient time-series plot showing the planetary orbits inclinations.

weather changes or social trends in newspapers or TV programs. Medical observations, such as blood pressure, heart beat rate or a body temperature changes is another example of time-series commonly seen in life. According to Tufte [61] “The time-series plot is the most frequently used form of graphic design. With one dimension marching along to the regular rhythm of seconds, minutes, hours, days, weeks, months, years, or millennia, the natural ordering of the time scale gives this design a strength and efficiency of interpretation found in no other graphic arrangement.” The figure 1.1 from the Tufte book depicts the oldest known example of a time-series plot showing the planetary orbits inclinations and is dated back to the tenth century.

It is generally assumed that consecutive measured values in a time-series are sampled at equally spaced time intervals (which is not always true and if required, resampling and interpolation can be used). This specific ordering of sample values in time often provides valuable information about the dependency of successive observations on earlier ones and is the key feature distinguishing time-series data from usually independent statistical samples. The explicit recognition of ordering in time-series analysis techniques makes them somewhat different from traditional statistical analyses [5]. It should be noted - that while many time-series analysis methods treat time-series as simple sequences of real numbers discarding the actual time-stamps, the time-ordering is always considered and it is assumed that unpronounced time interval are equal between the sampled values.

Time-series analysis methods in general combine various statistical and pattern recognition techniques while using information presented in the time-ordering of samples. Historically, time-series analysis is divided by two major fields of study: the first is explorative and descriptive analysis and the second is forecasting. While the descriptive analyses focus on the understanding of the time-series generating processes itself by finding trends, periodicity and some hidden features within the time-series [11], the predictive analyses aim at forecasting the future of the time-series

generating process based on the information found by conducting descriptive analyses [19]. Both - descriptive and predictive analyses are based on the identification of some pattern in the time-series which can be formally described and can correspond to the time-series generating process. Once this pattern is found and compiled into the formal model during the explorative analysis, it is used for extrapolation of the time-series into the future. Such model development and forecasting can be traced from the Babylonian astrologist predicting celestial phenomenas to the contemporary stochastic and deterministic models in many scientific fields.

As we mentioned before, time-series are ubiquitous. Taking into account the fact that most of the data collected automatically by sensing and monitoring are time series, it is hard not to overemphasize the importance and value of time-series data and analyses. Many public and private time-series databases exist for tracking and analysis of various information, and contemporary trends in the cost of data storage, increased bandwidth and progress in information science enable tremendous growth in time-series data volume and variety. There are public databases which track financial indexes and climate change (Figure 1.2), astronomical observations [31], medical information [42] and many others. This growth in temporal data volume and availability during the last decade of the twentieth century have created a demand for new approaches in KDD (Knowledge Discovery and Data mining) applications capable of handling very large volumes of temporal data. Since most KDD techniques such as classification, clustering and pattern discovery are based on the ability to quantify similarity between subjects, the need for advanced algorithms created a whole new research field dealing with data mining in temporal databases. Much research work has been done in the past 20 years, resulting in more than a dozen core algorithms which can be further extended for specific needs. In this work we will review the most relevant ones in order to understand their ability to perform in our application domain.

Fed System



**ECONOMIC RESEARCH**  
FEDERAL RESERVE BANK OF ST. LOUIS  
Advancing Economic Knowledge Through Research & Data

[RSS](#) | [Email Notifications](#) | [Data Lists](#) | [My Account](#) | [Log In](#)

[Publications](#) | [Economic Data - FRED®](#) | [Working Papers](#) | [Economists](#) | [Conferences](#) | [CREB®](#)

[Employment](#) | [Seminars](#) | [Monetary Aggregates](#)

Home > Economic Data - FRED®

## Economic Data - FRED®

[Categories](#) | [Sources](#) | [Releases](#) | [Updates](#) | [Series With Vintage Data](#) | [Download All Data](#) | [Notify Me of All Updates](#) | [Related Links](#) | [API](#) | [Help](#)

Welcome to FRED® (Federal Reserve Economic Data), a database of 20,056 U.S. economic time series. With FRED® you can download data in Microsoft Excel and text formats and view charts of data series. We plan to continually improve FRED® and encourage you to send feedback through our [contact form](#).

**Introducing FRED® API.** The [FRED® API](#) is a web service that allows developers to write programs and build applications that retrieve economic data from the [FRED®](#) and [ALFRED®](#) websites. Requests can be customized according to data source, release, category, series, and other preferences. [More...](#) (2009-03-16)


**Categories**

<a href="#">Banking</a> <a href="#">Business/Fiscal</a> <a href="#">Consumer Price Indexes (CPI)</a> <a href="#">Employment &amp; Population</a> <a href="#">Exchange Rates</a> <a href="#">Foreign Exchange Intervention</a> <a href="#">Gross Domestic Product (GDP) and Components</a>	<a href="#">Interest Rates</a> <a href="#">Monetary Aggregates</a> <a href="#">Producer Price Indexes (PPI)</a> <a href="#">Reserves and Monetary Base</a> <a href="#">U.S. Trade &amp; International Transactions</a> <a href="#">U.S. Financial Data</a> <a href="#">Regional Data</a>
---	--

**Updates**


[1-Month Certificate of Deposit: Secondary Market Rate](#) (2009-04-21)

[20-Year Treasury Constant Maturity Rate](#) (2009-04-21)



GCOS - AOPC/QOPC  
**Working Group on Surface Pressure**

You are at: [PSD Home](#) [WGSP Home](#)



**Climate Timeseries**

[Download](#)  
[Analyze & Plot](#)

**Gridded Datasets**

[Download](#)  
[Analyze & Plot](#)

**Group Business**

[Members](#)  
[Regional Coordinators](#)  
[Meeting Minutes](#)  
[Reports](#)

**Workshops**

[Upcoming](#)  
[Previous](#)

**Related Links**

---

**PSD Navigation:**  
[Home](#) | [Privacy Policy](#) | [Disclaimer](#) | [Contact](#)

## Overview

### Role

The role of the [Global Climate Observing System \(GCOS\)](#) Working Group on Surface Pressure (WG-SP) is to promote the development of long-term high-quality analyses of atmospheric surface pressure.

### Terms of Reference

The terms of reference of the GCOS Working Group on Surface Pressure are:

- to promote the analysis of global surface pressure from both real-time and historical sources using both daily and monthly data;
- to record and evaluate differences among surface pressure analyses through comparison of basic products;
- to recommend actions needed to ensure the quality and consistency of surface pressure analyses based on analysis of those differences;
- to promote the recovery of atmospheric pressure data, including issues associated with data access, archiving and maintenance;
- to promote the comparison of the various types of barometers and pressure sensors (including satellite estimates) used to measure surface pressure;
- to report annually to [Atmosphere Observation Panel for Climate \(AOPC\)](#) and [Ocean Observations Panel for Climate \(QOPC\)](#) on progress, recommendations and future plans of the Group.

The Working Group on Surface Pressure is organized under the auspices of the Global Climate Observing System. More information about GCOS can be found at its homepage: <http://www.wmo.int/pages/prog/gcos/index.php>

Figure 1.2: Examples of time-series databases. Upper screenshot is the FRED (Federal Reserve Economic Data) database of 20,056 U.S. economic time series, lower is the part of the NOAA Climate time-series database.

## Chapter 2

# Time-series similarity measurements

Though not present in classical time-series analyses such as trend and seasonality identification, forecasting and others, the time-series similarity<sup>1</sup> problem has become a cornerstone problem for recent data-mining applications. The ability to determine time-series similarity addresses many general time-series KDD problems:

- identifying stocks with similar movement in price [15] [6] [55]
- finding products with similar selling pattern [21]
- identifying music with a score similar to the copyrighted one [71] [56]
- performing speech to text conversion [2]
- verifying signatures and performing handwriting to text conversion [49] [43]
- finding objects with similar movement trajectories [69] [1] [64]
- detecting anomalies or events in signals [58] [70]
- finding developers with similar build patterns.

All of the solutions for this set of problems are based on the implementation of a time-series database enhanced with the ability to process “time-series similarity queries” or “range queries”.

First introduced by Goldin and Kanellakis [29], the concept of the “approximately similar time-series data” is based on “user-perceived similarity” rather than direct mathematical comparison with straightforward metrics. This very important observation essentially relaxes constraints for finding similar time-series. As an example of this relaxation, authors introduce scales and shifts in

---

<sup>1</sup>Note that a time-series (a finite sequence of real numbers) is alternatively called “sequence” or “signal” in the literature and we might use these terms depending on the context of the reviewed paper or method.



order to simulate human ability in comparing time-series features. Naming scale as the  $a$  coefficient and shift as  $b$  they introduced a form of the “similarity relation”  $T_{a,b}$  with properties:

1.  $\forall X, X = T_{0,1}(X)$ , Reflexivity or identity transformation (2.0.0.1)

2. if  $X = T_{a,b}(Y)$  then  $Y = T_{\frac{1}{a}, -\frac{b}{a}}(X) = T_{a,b}^{-1}(X)$ , Symmetry or inverse of  $T_{a,b}$  (2.0.0.2)

3. if  $X = T_{a,b}(Y)$  and  $Y = T_{c,d}(Z)$ , then  $X = T_{zc, ad+b}(Z) = (T_{a,b} * T_{c,d})(Z)$ , Transitivity (2.0.0.3)

It is natural to quantify this “similarity” by assigning a cost coefficient to each of the transformations. We will return to scale and shift transformations along with smoothing and matching envelope techniques in greater detail in the section 2.3. For now we focus on more general discussion of similarity and distance metrics properties.

Traditionally [3], time-series similarity queries are divided into two categories: whole sequence matching and subsequence matching where both categories are divided by first-occurrence and all-occurrences subproblems [29]. The Range Query problem of finding all (sub)sequences similar to a given one within the distance  $\epsilon$  is specific for this domain as well as the All-Pairs Query (“spatial joint”) of finding all pairs of (sub)sequences that are within  $\epsilon$  from each other [3]. While whole matching requires time-series to be exactly of the same length, subsequence matching considers the shorter query time-series for which it finds the best match in the longer template time-series using a sliding-window approach and reduces the task to the whole-sequence problem in each individual comparison. In both cases the similarity query mechanism relies on metrics with a well-defined distance function quantifying the time-series similarity.

The distance function on a set  $X$  defined as:

$$d : X \times X \rightarrow \mathbb{R} \tag{2.0.0.4}$$

And if  $x, y$  and  $z \in X$  the distance function  $d$  required to satisfy following conditions:

1.  $d(x, y) > 0$ , non-negativity (2.0.0.5)

2.  $d(x, y) = 0$ , if and only if  $x = y$  identity (2.0.0.6)

3.  $d(x, y) = d(y, x)$ , symmetry (2.0.0.7)

4.  $d(x, z) \leq d(x, y) + d(y, z)$ , the triangle inequality (2.0.0.8)

There are number of functions which can be used in the time-series similarity domain and we discuss some of them: Euclidean distance in section 2.1, mentioned “transformation rules” in section 2.3, Dynamic Time Warping (DTW) in section 2.4 and Longest Common Subsequence (LCS) in section 2.5.

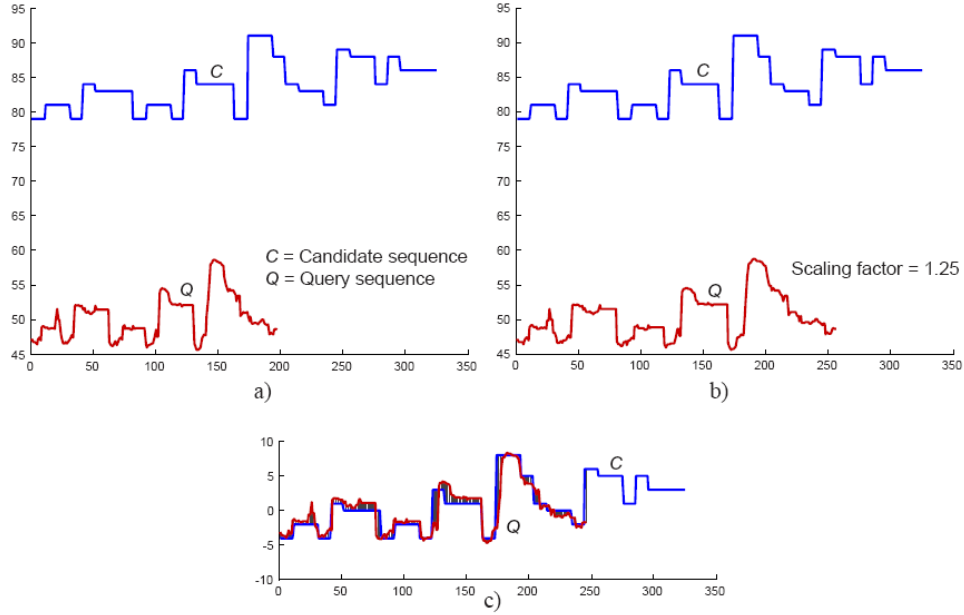


Figure 2.1: a) Figure taken from [25] depicts raw pitch contour extracted from a sung query represents a query sequence  $Q$ , and a MIDI pitch contour of Happy Birthday song represents a candidate sequence  $C$ . b) A rescaled query sequence  $Q$  with scaling factor = 1.25. c) Both sequences after mean normalization at the queries length. The shaded region shows their Euclidean distance.

## 2.1 Euclidean distance

Given two time-series  $X = (x_1, x_2, \dots, x_n)$  and  $Y = (y_1, y_2, \dots, y_m)$  where  $n = m$ , the Euclidean distance is defined as

$$D_{euclidean}(X, Y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (2.1.0.9)$$

While being the easiest to calculate and satisfying all of the distance requirements, the Euclidean distance, when applied to raw time-series data, seems to be too rigid to capture similarities between two time-series following the very same pattern in time but differ in scale. In [25] authors explain that Euclidean distance is “...impractical in several applications, particularly for multimedia applications, where shrinking and stretching of the data are very typical” and provide an example of two time-series representing the “Happy birthday” tune which were found distinct by the direct application of the Euclidean distance, but after normalization and scaling appeared to be identical as shown in Figure 2.1.

The Euclidean distance has a linear complexity.

## 2.2 Time series normalization

Normalization is a type of mathematical transformation of time series from one value domain into another value domain with the purpose of obtaining a specific set of statistical features such as a limit of values, a certain variance, standard deviation, or average for the transformed (normalized) time series. The operation usually takes one or two steps and transforms each of the elements  $x_i \in X$  of input sequence into the element  $x'_i \in X'$ , where  $X'$  is a normalized sequence. There are different types of normalization such as:

- Normalization into an Interval [51] [12]
- Normalization to Sum 1
- Normalization to Euclidean Norm 1
- Normalization to Zero Mean
- Normalization to Zero Mean and Unit Standard Deviation [29]

### 2.2.1 Normalization into an Interval

This type of normalization procedure ensures that all elements of an input vector are scaled proportionally into an output vector with predefined upper and lower limits. Let's  $L_{min}$  to be a desired lower limit,  $L_{max}$  to be a upper limit and let's  $x_{max} = \max\{x_i, x_i \in X\}$  and  $x_{min} = \min\{x_i, x_i \in X\}$ . Than

$$x'_i = \frac{(x_i - x_{min})(L_{max} - L_{min})}{x_{max} - x_{min}} + L_{min}, i \in \mathbb{N} \quad (2.2.1.1)$$

is a normalization procedure.

### 2.2.2 Normalization to Sum 1 and Normalization to Euclidean Norm 1

While normalization to Sum 1 ensures that elements of  $X'$  sum up to 1:

$$1 = \sum_{i=1}^N x'_i \quad (2.2.2.1)$$

by normalization procedure:

$$x'_i = \frac{x_i}{\sum_{i=1}^N x_i} \quad (2.2.2.2)$$

the Normalization to Euclidean Norm 1 transforms the input vector values proportionally into an output vector with a Euclidean norm of 1:

$$1 = \sum_{i=1}^N x_i'^2 \quad (2.2.2.3)$$

by transformation procedure:

$$x_i' = \frac{x_i}{\sum_{i=1}^N x_i^2} \quad (2.2.2.4)$$

### 2.2.3 Normalization to the zero mean

This method ensures that the mean of the normalized vector will be approximately 0. The mean of the vector  $X$  of length  $N$  calculated as:

$$\mu_X = \frac{1}{N} \sum_{i=1}^N x_i \quad (2.2.3.1)$$

and the normalization procedure is:

$$x_i' = x_i - \mu, \quad i \in \mathbb{N} \quad (2.2.3.2)$$

The time-series shown at the Figure 2.1 are normalized to zero mean.

### 2.2.4 Normalization to Zero Mean and Unit Standard Deviation

This type of normalization is also called “Normalization to Zero Mean and Unit of Energy” in research literature and first found in [29]. It ensures that all elements of the input vector are transformed into the output vector whose mean is approximately 0 while the standard deviation (and variance) are in a range close to 1. This procedure uses mean  $\mu$  and standard deviation which calculated as

$$\sigma = \sqrt{\frac{\sum_{i=1}^N (x_i - \mu)^2}{N - 1}} \quad (2.2.4.1)$$

or equivalently

$$\sigma = \sqrt{\frac{N \left( \sum_{i=1}^N x_i^2 \right) - \left( \sum_{i=1}^N x_i \right)^2}{N(N - 1)}} \quad (2.2.4.2)$$

The normalization itself is

$$x_i' = \frac{x_i - \mu}{\sigma}, \quad i \in \mathbb{N} \quad (2.2.4.3)$$

and yields the vector  $X_i'$  such as  $\mu_{X'} \approx 0$  and  $\sigma_{X'} \approx 1$ .

According to most of the recent works [29] [47] [48] this type of time-series normalization is

the best known transformation of the raw time-series which preserves original time-series features. Nevertheless, the article by Lin et al. [47] explains the reasons and solutions for some cases when the zero mean and unit standard deviation normalization fails. For example if a signal is constant over most of the time span with minor noise at short intervals, this normalization will overamplify the noise to the maximal amplitude. This will also occur if the time-series contains only single value and the standard deviation is not defined.

## 2.3 Transformation rules (smoothing, scales and shifts)

As we have seen, the Euclidean distance is efficient and easily computable but it doesn't capture similar time-series in the presence of noise, scales or shifts. Much effort has been put in to overcome this issue. As mentioned before, in [29] authors introduced scale and shift shape-perceiving transformations which are applied before measuring the similarity with Euclidean distance. Their proposed definition of similar time-series is based on the transformation  $T_{a,b}$  where  $a$  is the scaling coefficient and  $b$  is the shift coefficient. This "similarity transformation  $T_{a,b}$ " is a mapping of each of the  $x_i \in X$  into  $x'_i = a * x_i + b$  and if this transformation can be found for two time-series  $X$  and  $Y$  such that  $X = T_{a,b}(Y)$  time-series  $X$  and  $Y$  are considered to be similar.

Agrawal et al. [4] proposed a more general method of aligning of time-series by allowing any arbitrary segment of the query time-series to be scaled and stretched by any suitable amount and adding the ability to delete any non-matching segment (note that this work also seems to be the one of the first to propose LCS application to time-series similarity, see section 2.5). Also, they introduced an  $\epsilon$  envelope around the template sequence which aimed to deal with noise on the query time-series. Figure 2.2 shows the principles of such a method working from the raw time series through discussed transformations.

Since the Euclidean distance is calculated only after the discussed transformations are applied, it is unable to capture the cost and complexity of the these transformations and does not reflect the true similarity. But if the cost is assigned to the each of the transformations performed [37], the integral of all effective costs will measure the true similarity between time-series.

The complexity of scales and shift transformation is  $O(NM)$  where  $N$  and  $M$  are lengths of the query and template sequences.

## 2.4 Dynamic Time Warping (DTW)

Another commonly used metric<sup>2</sup> for computing time-series similarity is based on the Dynamic Time Warping algorithm (DTW). The DTW algorithm is a well-known algorithm with applications to many areas. While first introduced in 60's [9] and extensively explored in 70's for application to

---

<sup>2</sup>It should be noted, that while a distance function is required to satisfy 2.0.0.5 - 2.0.0.8, the Dynamic Time Warping (DTW) distance fails to satisfy the triangular inequality 2.0.0.8 as shown in [63] and [50].

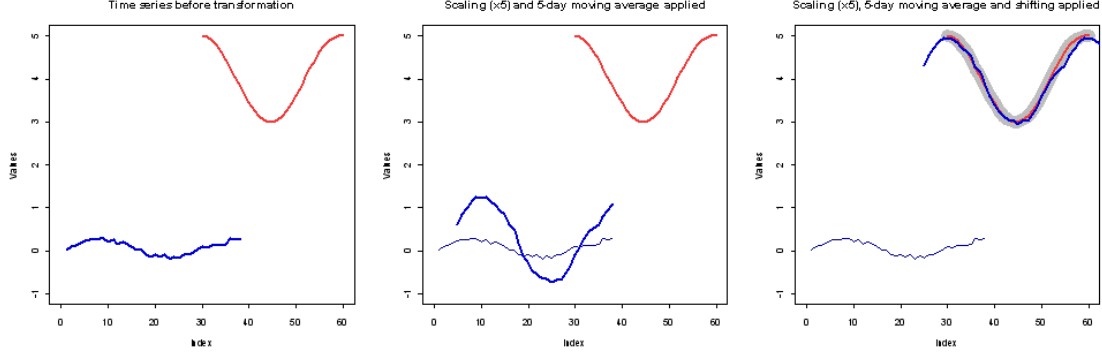


Figure 2.2: Illustration of scaling, smoothening and shifting: the leftmost plot depicts the raw time-series, the plot at the middle shows the query time-series after scaling and smoothening with 5-day moving average, and the plot at the right adds shifts to transform while showing  $\epsilon$  alignment envelope.

speech recognition [53], [59] it is currently used in many areas: handwriting and online signature matching [24] [60], sign language recognition [44] and gestures recognition [18] [44], data mining and time series clustering (time series databases search) [56] [30] [7] [34] [35] [26], computer vision and computer animation [1], surveillance [69], protein sequence alignment and chemical engineering [62], music and signal processing [52] [1] [2].

The idea behind the DTW algorithm lies in allowing shifts of the data points in time (warping) and relies on the usage of Dynamic Programming [9] for optimal time-series alignment. The algorithm starts by building the distance matrix  $C \in \mathbb{R}^{N \times M}$  representing all pairwise distances between  $X$  and  $Y$ . This distance matrix called the **local cost matrix**  $C_l$  for the alignment of two sequences  $X$  and  $Y$ :

$$C_l \in \mathbb{R}^{N \times M} : c_{i,j} = \|x_i - y_j\|, i \in [1 : N], j \in [1 : M] \quad (2.4.0.4)$$

Once the local cost matrix is built, the algorithm finds the **alignment path** which runs through the low-cost areas - “valleys” on the cost matrix, Figure 2.3. This alignment path (or **warping path**, or **warping function**) defines the correspondence of an element  $x_i \in X$  to  $y_j \in Y$  following the boundary condition which assigns first and last elements of  $X$  and  $Y$  to each other, Figure 2.3.

Formally speaking, the alignment path built by DTW is a sequence of points  $p = (p_1, p_2, \dots, p_K)$  with  $p_l = (p_i, p_j) \in [1 : N] \times [1 : M]$  for  $l \in [1 : K]$  which must satisfy to the following criteria:

1. **Boundary condition:**  $p_1 = (1, 1)$  and  $p_K = (N, M)$ . The starting and ending points of the warping path must be the first and the last points of aligned sequences.
2. **Monotonicity condition:**  $n_1 \leq n_2 \leq \dots \leq n_K$  and  $m_1 \leq m_2 \leq \dots \leq m_K$ . This condition

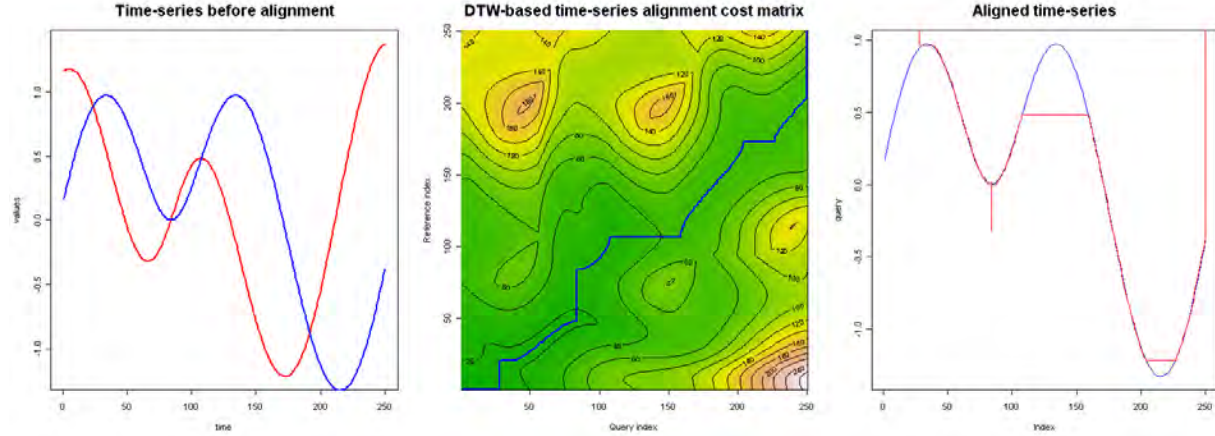


Figure 2.3: Illustration of the DTW algorithm. Left: the raw time-series; center: the alignment matrix and the optimal alignment path; right: aligned time-series.

preserves the time-ordering of points.

3. **Step size condition (continuity)**: this criteria limits the warping path from long jumps (shifts in time) while aligning sequences. The basic step size condition formulated as  $p_l \in \{(1, 1), (1, 0), (0, 1)\}$ . Note that numerous research [59] [33] [53] [54] experiments with this condition revealed its significant impact on the algorithm performance and various step patterns were proposed.

The **cost function** associated with a warping path computed with respect to the local cost matrix (which represents all pairwise distances) is:

$$c_p(X, Y) = \sum_{l=1}^L c(x_{n_l}, y_{m_l}) \quad (2.4.0.5)$$

The warping path which has a minimal cost associated with alignment is called the **optimal warping path**  $P^*$ . Once computed, it is used for time-series alignment as shown at the Figure 2.3. The Euclidean Distance is usually applied after warping in order to quantify the similarity.

It's quite interesting that Euclidean distance itself is the special case of the DTW distance with restriction that  $i = j = k$ .

The time and space complexity of the DTW distance is  $O(NM)$

## 2.5 Longest Common Subsequence (LCS)

Another widely used metric in computing time-series similarity is the Longest Common Subsequence or LCS. LCS could be seen as the successor of the application of the Euclidean distance, DTW or scales and shifts [4] since it can be based on any of these. The idea of LCS is explained in the

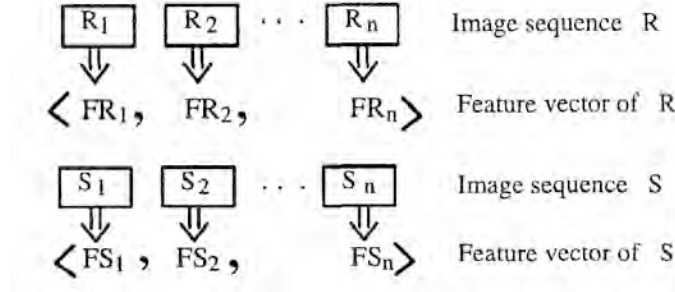


Figure 2.4: This figure is taken from [66] and while it designed to explain LCS application to the sequences of images it easily explains any of LCS and feature decomposition based approach to time-series similarity.

classical Computer Science manuscript [17] (CLR) for the string matches with use of the Dynamic Programming and has the complexity of  $O(NM)$ .

Yazdani and Ozsoyoglu [66] proposed a new algorithm with better complexity  $O(N+M)$  specifically for image sequences. They also generalized it further for any real-valued signals and specifically for application to time-series representing sets of features like Fourier coefficients, etc. Figure 2.4 taken from the article explains the approach taken: each of the images is approximated by the set of real-valued features, and if  $R$  and  $S$  match then  $FR$  and  $FS$  match analogously, if  $FR$  and  $FS$  are similar then  $R$  and  $S$  are similar.

The algorithm starts with two sequences  $X$  and  $Y$  with lengths  $N$  and  $M$  respectively. Then, it builds vector data structure  $C_{vect}$  of the size of the shorter one such that  $C_{vect}[i]$  is the index of the element in  $Y$  which matches with  $i$ th element of  $X$ . By letting  $C[i, j]$  to be the length of the longest common subsequence of  $X_i$  and  $Y_j$ , they define it as

$$C[i, j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ C[i - 1, j - 1] + 1 & \text{if } i, j > 0 \text{ and } x_i \text{ matches } y_j \\ \max C[i, j - 1], C[i - 1, j] & \text{otherwise} \end{cases} \quad (2.5.0.6)$$

By modifying the range of matching in the definition 2.5.0.6 to the specific (relaxed) bounds they achieved a better performance as well as being able to improve the sensitivity of the method.

Agrawal et al. [4], as mentioned before, introduced an LCS-based concept of similarity with local scaling, shifting and deletions, measuring the similarity based on the amount of non-overlapping time-ordered pairs of subsequences that are similar instead of individual points as displayed at the Figure 2.5. The complexity of LCS algorithm is  $O(NM)$ .



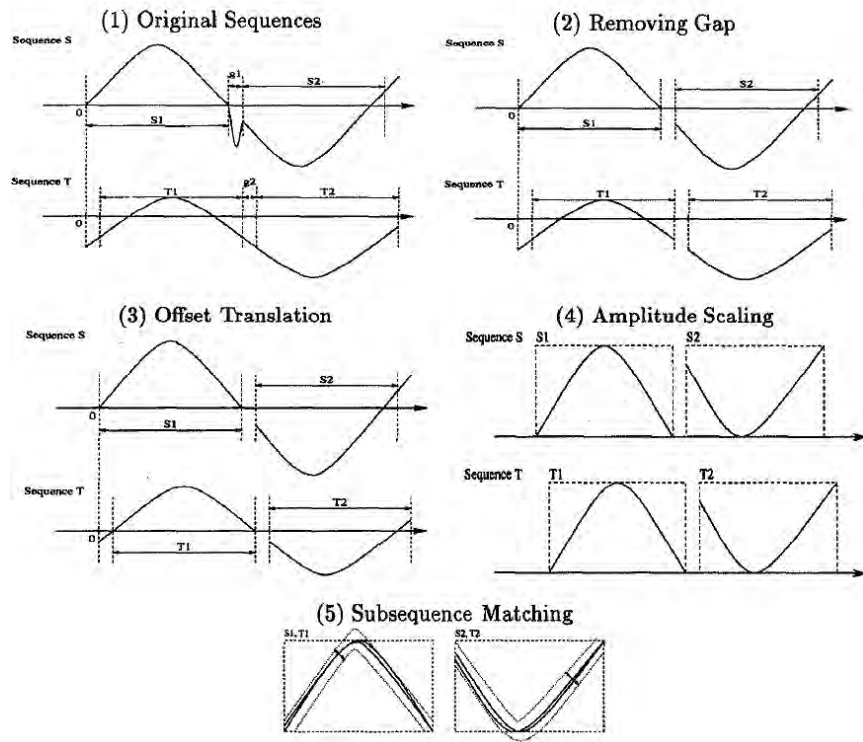


Figure 2.5: This figure is taken from [4] and explains the modified LCS application for the sequence matching.

## Chapter 3

# Dimensionality reduction, indexing and bounding

As we saw with the application of LCS to images in [66], Yazdani and Ozsoyoglu characterized each of the images by a set of Fourier coefficients. By employing this approach they managed to resolve the “dimensionality curse” in the time series-similarity problem which essentially decreases the performance of any similarity algorithm to the one of the sequential search with the growth of dimensionality above 16 [32] [68] [10] [13] [8].

The problem addressed in this chapter is the approximation (lossy compression) of a time series data in a way which maintains fast random access, comparison and indexing of time-series with minimal errors. There are many approximations proposed in the research literature and Figure 3.1 from [47] illustrates them in a hierarchy. All of the methods essentially consist of two steps: at first they transform a given set of time series into some low-dimensional representation and secondly index them in order to utilize data-mining algorithms later. Important issues to understand include the sensitivity and selectivity of algorithms, their complexity and performance.

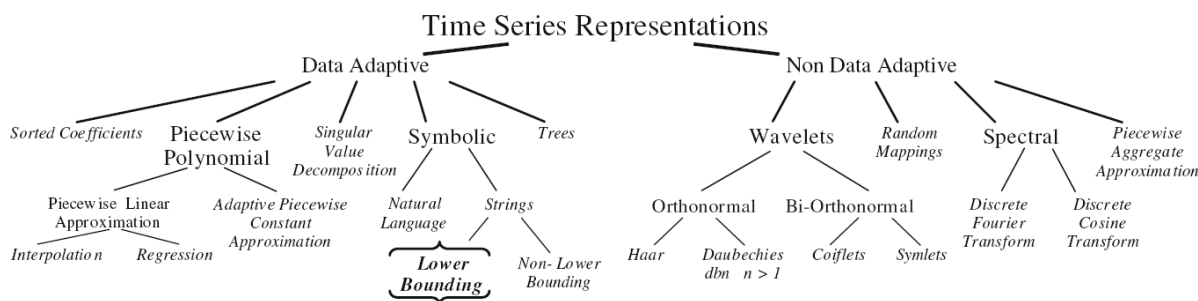


Figure 3.1: The figure from [47] illustrating a hierarchy of all of the flavors of time series representation (decomposition).

### 3.1 Lower Bounding

The first approach to time-series indexing by Agrawal et al [3] employed DFT for dimensionality reduction and F-index for similarity search. The proposed time-series approximation with only first  $c$  “significant” DFT coefficients resulted in false hits when searching F-index, but it was shown that all false hits are false-positive and there are no false-dismissals. The author’s proof of algorithm correctness based on the “contractive property” (lower-bounding condition) of their transform and feature-space distance choice. The proof explains that the distance in the feature space essentially “underestimates” the real distance between time-series.

The lower bounding condition is formulated as:

$$D_{feature}(T(X), T(Y)) \leq D_{object}(X, Y) \quad (3.1.0.1)$$

where  $D_{feature}$  is the distance between objects in feature space (Agrawal et al used the Euclidean distance),  $T$  is the transform function from object space to feature space (DFT) and  $D_{object}$  is the real distance between objects  $X$  and  $Y$  (the Euclidean distance again).

Agrawal et al has shown that lower-bounding condition holds for DFT transform by relying on Parseval’s theorem which guaranteed that the distance between two time-series in the time domain is the same as in the frequency domain (3.2.0.5) and the dismissal of some DFT coefficients only decreases the distance value in the feature space. Subsequent work by Faloutsos et al [28] formalized the lower-bounding condition requirement for the range queries processing without false-dismissals.

### 3.2 Discrete Fourier Transform (DFT) based decomposition

Agrawal et al. in [3] proposed a DFT-based method for the time-series indexing and similarity queries processing. Their idea is based on the observation of a fairly good approximation of the time-series by only few “strong” frequencies and on Parseval’s Theorem (aka Rayleigh’s energy theorem).

For each of the time-series  $x$  of length  $N$ , the authors propose to extract only the first  $c$  DFT coefficients (frequencies), where  $c \ll N$  and  $f_c$  is the “cutoff frequency”. Thus each of the time-series is mapped into the low  $c$  dimensional space and stored in the  $F$ -index, where the search over the index is implemented by using an  $R^*$  tree [8]. The authors argue that the approach taken is characterized by the “completeness of the feature-extraction” and it is “efficiently dealing with the dimensionality curse”.

The  $n$ -point DFT transform of the time-series  $X = (x_0, x_1, \dots, x_{n-1})$  is defined as:

$$X_f = 1/\sqrt{(n)} \sum_{t=0}^{n-1} x_t \exp(-j2\pi ft/n), \quad t = 0, 1, \dots, n-1, \quad j = \sqrt{-1} \quad (3.2.0.2)$$

also the inverse transform:

$$x_t = 1/\sqrt{(n)} \sum_{f=0}^{n-1} X_f \exp(j2\pi ft/n), \quad t = 0, 1, \dots, n-1 \quad (3.2.0.3)$$

and the fundamental observation of the Parseval's theorem is

$$\sum_{t=0}^{n-1} |x_t|^2 = \sum_{f=0}^{n-1} |X_f|^2 \quad (3.2.0.4)$$

that the energy of the sequence in the time domain equals the energy in the frequency domain.

Furthermore, the authors show that Discrete Fourier Transform inherits linearity and preservation of amplitude coefficients during the time-shifts from the Continuous Fourier Transform. Taking these properties and Parseval's theorem in account, the authors show that

$$\|x - y\|^2 \equiv \|X - Y\|^2 \quad (3.2.0.5)$$

which implies the equivalence of the Euclidean distances between two sequences  $x$  and  $y$  in the time and frequency domains.

Agrawal et al. introduces the distance function  $D$  where the distance between two sequences  $x$  and  $y$  is the square root of the energy of the difference:

$$D(x, y) \equiv \left( \sum_{t=0}^{n-1} |x_t - y_t|^2 \right)^{1/2} \equiv (E(x - y))^{1/2} \quad (3.2.0.6)$$

where  $E(x)$  is the energy of the sequence:

$$E(x) \equiv \|x\|^2 \equiv \sum_{t=0}^{n-1} |x_t|^2 \quad (3.2.0.7)$$

and implements a similarity relation between two sequences by using a user defined threshold  $\epsilon$  - i.e. if the distance between two sequences falls below threshold they considered to be similar:

$$D(x, y) \leq \epsilon \Rightarrow x \text{ similar to } y \quad (3.2.0.8)$$

Continuing the discussion, the authors compare their DFT-based method to the “sequential scanning” method (the only alternative available by that time). They use the same  $R^*$  tree for the sequential indexing and implement an “early abandoning”: as soon as the calculated Euclidean

distance exceeds  $\epsilon^2$ , two sequence declared dissimilar. It was shown by comparing two approaches, that with the growth of the dataset volume and the length of sequences, the DFT-based approach outperforms the sequential search. Another interesting point shown is that, in general,  $f_c \approx 3$  is enough to capture time-series features and build an index which provides satisfactory performance and has a contractive property, i.e. ensures no false-dismissal.

### 3.3 Singular Value Decomposition (SVD)

Singular Value Decomposition for ad-hoc query support in time-series databases was proposed by Korn et. al. in [42]. They start with a case study of a typical warehouse dataset (AT&T) of sale patterns and represent this data as a matrix  $X$  of size  $N \times M$  with rows corresponding to distinct customers and columns to the time intervals. One of the observations is that the amount of customers is much larger than the length of the time-series:  $N \gg M$ . Another observation is that the size of such a matrix could easily become unmanageable if one must process “random access” queries. The proposed solution for this problem is to design a “compression” algorithm which trades off some of the precision in order to allow processing of significant amounts of data with random access queries.

While being aware of some convenient compression techniques such as LZ and Huffman, as well as the spectral decomposition (DFT, DCT and DWT) and clustering techniques, Korn et. al. are arguing that their approach with Singular Value Decomposition is the only solution which does not suffer from CPU-time complexity performance boundaries and performs spectral decomposition in an optimal manner.

The SVD decomposition method is based on a theorem stating that the real-valued matrix  $X$  can be expressed as

$$X = U \times \Lambda \times V^t \quad (3.3.0.9)$$

where  $U$  is a column-orthonormal  $N \times r$  matrix ( $r$  is rank of the matrix  $X$ ),  $\Lambda$  is a diagonal  $r \times r$  matrix of eigenvalues  $\lambda_i$  of  $X$  and  $V$  is a column-orthonormal  $M \times r$  matrix. Which essentially is the spectral decomposition:

$$X = \lambda_1 u_1 \times v_1^t + \lambda_2 u_2 \times v_2^t + \dots + \lambda_r u_r \times v_r^t \quad (3.3.0.10)$$

of the matrix  $X$  where  $u_i$  and  $v_i$  are column vectors of the  $U$  and  $V$  and  $\lambda_i$  is a diagonal elements of  $\Lambda$ .

Once  $X$  is decomposed (two passes over the matrix) into the form of spectrum 3.3.0.10, the authors suggest to truncate the sum in equation to the first  $k$  terms,  $k \leq r \leq M$ :

$$\hat{X} = \sum_{i=1}^k \lambda_i u_i \times v_i \quad (3.3.0.11)$$

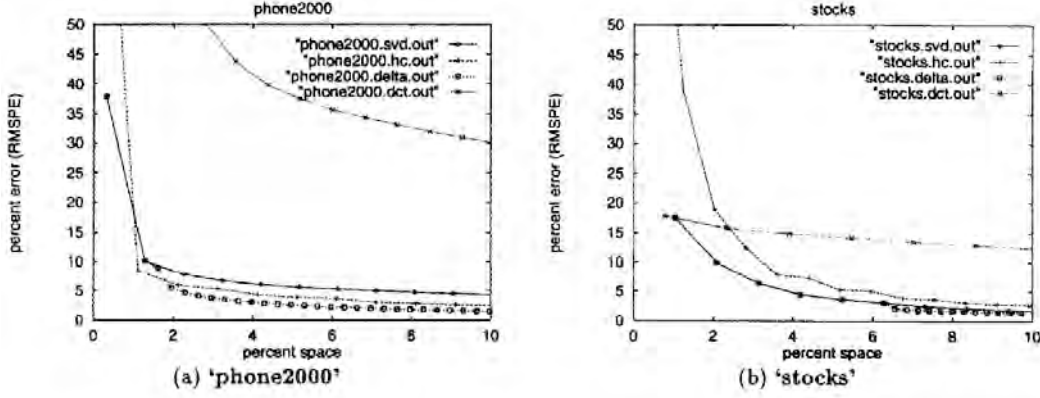


Figure 3.2: The Reconstruction Error (RMSPE) versus disk storage space (%) for clustering, DCT, SVD and SVDD.

where  $k$  depends on the space restriction. Note, that these  $k$  terms also known as the “principal components” in the PCA analysis [23].

It is shown that the compression ratio after applying the SVD transform is

$$s = \frac{N * k + k + k + k * M}{N * M} \approx \frac{k}{M} \quad (3.3.0.12)$$

since  $N \gg M \geq k$ . and the reconstruction of the any cell of the original matrix  $\hat{X}$  takes only  $O(k)$  time:

$$\hat{x}_{i,j} = \sum_{m=1}^k \lambda_m * u_{i,m} * v_{j,m}, \quad i = 1, \dots, N; \quad j = 1, \dots, M \quad (3.3.0.13)$$

Since the SVD approximation could yield considerable errors when restoring some of the original data, authors suggest keeping correcting information in the form of tuples (*row, column, delta*) for the values which approximate poorly (outliers), calling this method as “SVD with Details” or SVDD.

With respect to the performance of both SVD and SVDD methods, they were compared with DCT (Discrete Cosine Transform, which was shown to be more efficient than DFT [41], [22]) and the S-PLUS embedded clustering method and found to be significantly superior (Figure 3.2), especially SVDD, with the growth of the space allowed to use for corrective information storage.

### 3.4 Discrete Wavelet Transform (DWT) decomposition

In [57] Popivanov and Miller discuss time-series similarity search using wavelets-based dimensionality reduction. In this work, they move away from previously used Haar wavelets [15] and show that their wavelets outperform not only Haar, but also the best-known non-wavelet approaches for

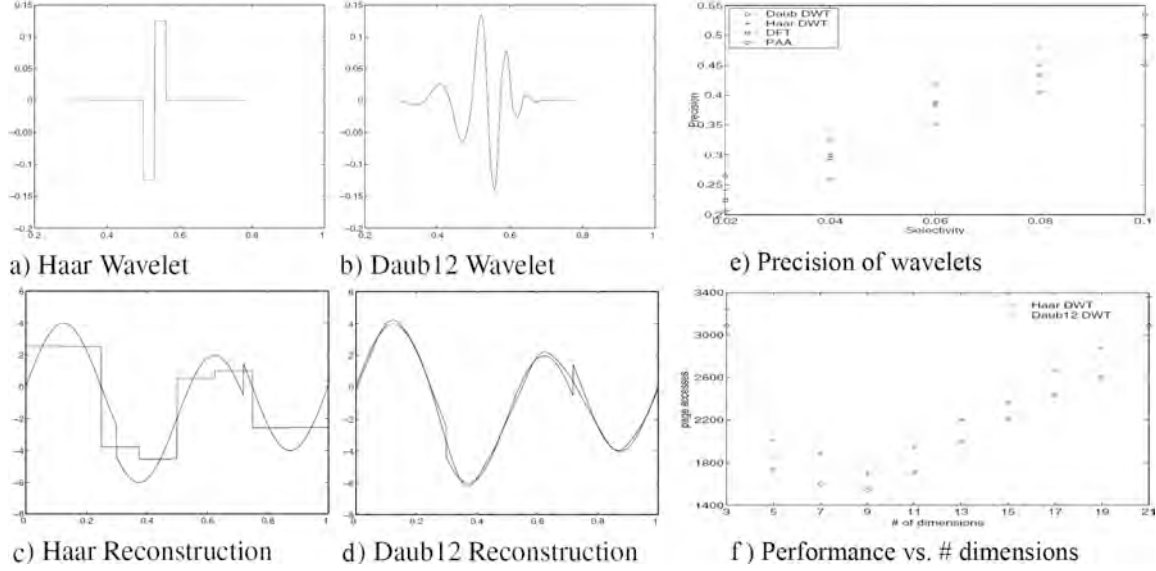


Figure 3.3: The combination of figures from [57] illustrating an advantage of Daubechies wavelet over the Haar wavelet (a, b, c, d, f) and other methods (e) such as DFT and PAA.

the time-series similarity search up to date (DFT, DCT, SVD etc.). Their approach for time series approximation benefits from such a wavelet-transform properties as:

- Compact support: there are flavors of the wavelet transform such that a basis function is non-zero only at the finite interval, which allows wavelet approximation to capture local time-series features as opposite to DFT based approach which is capturing only global features.
- Efficiency: while FFT complexity is  $O(n \log n)$ , wavelet transform is linear to the size of the data and is real-to-real transform.
- Sensitivity: the wavelet transform allows fine tuning of the approximation by selecting the different scales, corresponding to basis functions of different length where the DFT gives.

The wavelet is a function,  $\psi_{j,k}$  defined on  $\mathbb{R}$  with shift  $k$  and dyadic dilation (a product of power of 2, referred as stretching). The set of functions defined as

$$\psi_{j,k} = 2^{j/2} \psi(2^j t - k) \quad (3.4.0.14)$$

for  $j, k \in \mathbb{Z}$  form a complete orthonormal system over  $L^2(\mathbb{R})$  and can be used to define any signal  $f \in L^2(\mathbb{R})$  uniquely by the following series:

$$f = \sum_{j,k \in \mathbb{Z}} \langle f, \psi_{j,k} \rangle \psi_{j,k}, \text{ where } \langle f, g \rangle = \int_{\mathbb{R}} f g \, dx, \text{ the inner product} \quad (3.4.0.15)$$

The function  $\psi_{j,k}$  is the “basis function” of the wavelet (“mother wavelet” function) and essentially the wavelet can use an infinite family of basis functions as opposed to Fourier transform which uses only exponential function.

It has been proven [20] that the energy of the wavelet features lies between two non-negative bounds:

$$A \|X\|^2 \leq \|x\|^2 \leq B \|X\|^2 \quad (3.4.0.16)$$

where  $A$  and  $B$  are non-negative constants,  $x$  is the signal and  $X$  it’s wavelet transform  $X = T(x)$ .

For the  $\epsilon$ -radius range query and equation 3.4.0.16 we have the following inequality:

$$A \|X - Q\|^2 \leq \|x - q\|^2 < \epsilon^2 \quad (3.4.0.17)$$

which provides us with the confidence of no false-dismissals (lower bounding property).

Popivanov & Miller along with Chan & Fu [15] proved the superiority of DWT over FFT, SVD and PAA methods in both, the speed of index building and the precision of the range-query processing while using the same F-index and  $R^*$ -tree as in the previous time-series similarity work.

### 3.5 Piecewise Approximation of time-series (PLA, PCA, PAA, APCA)

Following the first wave of time-series similarity methods based on the spectral decomposition such as DFT, DCT, SVD, CHEB<sup>1</sup> and Haar wavelets, another approach has become popular among the time-series data-mining community: piecewise approximation. Faloutsos et al [27] proposed Piecewise Flat Approximation, Morinaka et al [51] and Chen et al [16] proposed Piecewise Linear Approximation (PLA), Yi & Faloutsos [67] and Keogh et al [38] followed with Piecewise Aggregate Approximation (PAA), Chakrabarti et al [14] proposed Adaptive Piecewise Linear Approximation (APLA). All this work has shown that surprisingly simple piecewise-based approximation methods outperform previous spectral decomposition based techniques by being easy to compute and index while satisfying the contractive property condition (3.1.0.1).

We will review the PAA method which approximates the time-series  $X$  of length  $n$  into vector  $\bar{X} = (\bar{x}_1, \dots, \bar{x}_M)$  of any arbitrary length length  $M \leq n$  where each of  $\bar{x}_i$  is calculated by following the next formula:

$$\bar{x}_i = \frac{M}{n} \sum_{j=n/M(i-1)+1}^{(n/M)i} x_j \quad (3.5.0.18)$$

This simply means that in order to reduce the dimensionality from  $n$  to  $M$ , we first divide the original time-series into  $M$  equally sized frames and secondly compute the mean values for each

---

<sup>1</sup>Chebyshev polynomials based decomposition methods were omitted in this review due to the space constraints and could be found in [12] and overall very similar to APCA in performance



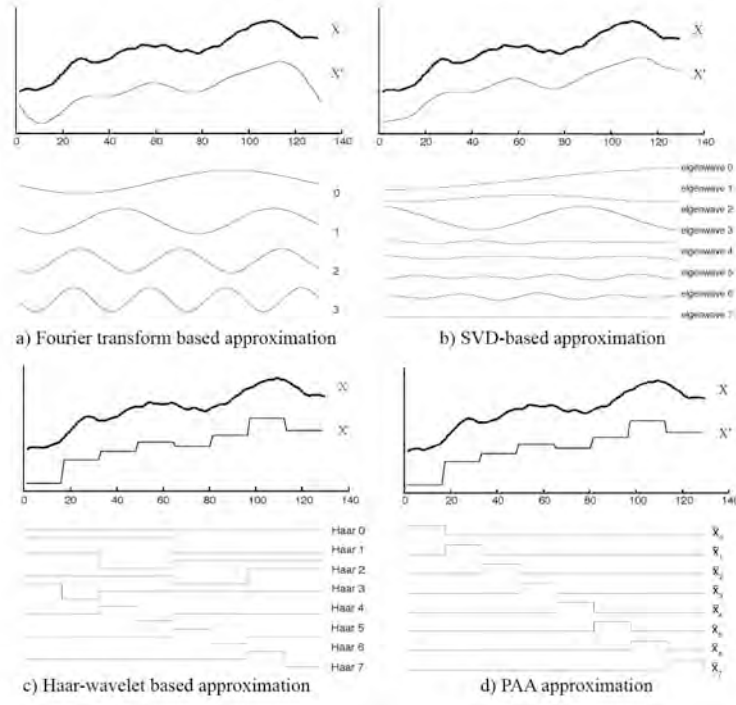


Figure 3.4: The combination of figures from [38] depicts different approaches for the time-series approximation (decomposition): a) the time-series spectral approximation (Fourier); b) SVD-based approximation; c) Haar-wavelet based approximation; d) Piecewise Aggregate Approximation where transformed values shown as “box” basis functions.

frame. The sequence assembled from the mean values is the PAA transform of the original time-series. It was shown by Keogh et al that the complexity of the PAA transform can be reduced from  $O(NM)$  (3.5.0.18) to  $O(Mm)$  where  $m$  is the number of sliding windows (frames). The satisfaction of the transform to bounding condition in order to guarantee no false dismissals was also shown by Yi & Faloutsos and Keogh et al by introducing the distance:

$$D_{PAA}(\bar{X}, \bar{Y}) \equiv \sqrt{\frac{n}{M}} \sqrt{\sum_{i=1}^M (\bar{x}_i - \bar{y}_i)^2} \quad (3.5.0.19)$$

and showing that  $D_{PAA}(\bar{X}, \bar{Y}) \leq D(X, Y)$ .

Concluding the piecewise based time-series approximation methods review we should note that PAA is very similar to Haar-wavelet based approach [15] as shown at Figure 3.4. Another nice feature of the PAA is the ability to process range queries with sequences of unequal to index size dimension, as was shown by Keogh et al in [38].

Overall, the PAA based approach to the time-series similarity problem was found to be very competitive in precision to the spectral-decomposition based methods while outperforming all com-

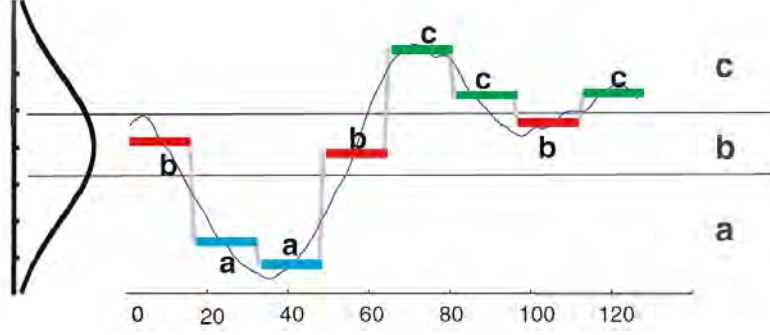


Figure 3.5: The illustration of the SAX approach taken from [47] depicts two pre-determined breakpoints for the three-symbols alphabet and the conversion of the time-series of length  $n = 128$  into PAA representation followed by mapping of the PAA coefficients into SAX symbols with  $w = 8$  and  $a = 3$  resulting in the string **baabccbc**.

petitors in the speed of index building. In terms of the index performance, PAA index has a constant time of insertions and deletion whether in case of SVD, for example, we have to rebuild the whole matrix. Also, Keogh et al has shown that PAA has ability to handle weighted Euclidean distance metrics, allowing implementation of more sophisticated querying techniques like “relevance feedback” [40].

### 3.6 Symbolic Aggregate approXimation (SAX)

The last approach to the time-series similarity problem that we shall review here is the current state of the art in time-series representation and dimensionality reduction method called Symbolic Aggregate approXimation (SAX). This approach transforms the original time-series data into symbolic strings.

Symbolic Aggregate approXimation was proposed by Lin et al in [47]. This method extends PAA-based approach inheriting algorithm simplicity and low computational complexity while providing satisfiable sensitivity and selectivity in range-query processing. Moreover, the use of a symbolic representation opens the door to the existing wealth of data-structures and string-manipulation algorithms in computer science such as hashing, regular expression pattern matching, suffix trees etc.

SAX transforms a time-series  $X$  of length  $n$  into the string of arbitrary length  $\omega$ , where  $\omega \ll n$  typically, using an alphabet  $A$  of size  $a \geq 2$ . The SAX algorithm consist of two steps: during the first step it transforms the original time-series into a PAA representation and this intermediate representation gets converted into a string during the second step. Use of PAA at the first step brings the advantage of a simple and efficient dimensionality reduction while providing the important lower bounding property as shown in the previous section. The second step, actual conversion of PAA coefficients into letters, is also computationally efficient and the contractive property of

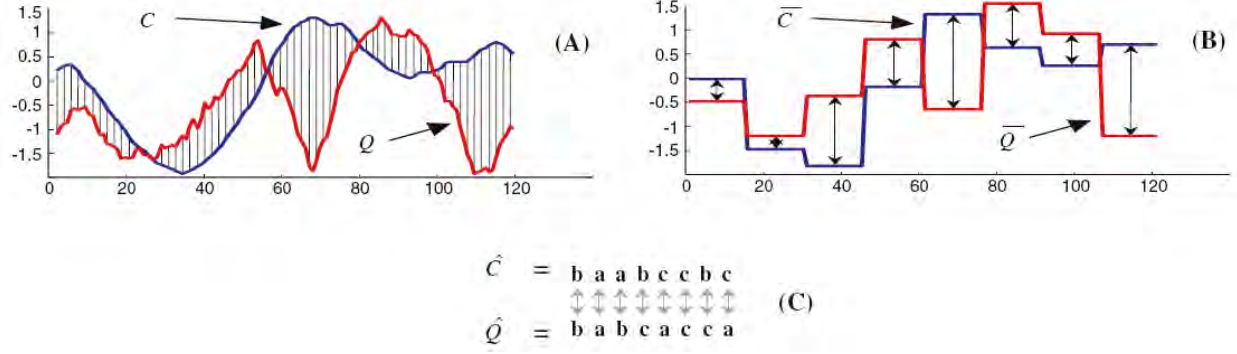


Figure 3.6: The visual representation of the two time-series  $Q$  and  $C$  and three distances between their representation: Euclidean distance between raw time-series (A), the distance defined for PAA coefficients (B) and the distance between two SAX representations (C).

symbolic distance was proven by Lin et al in [46].

Discretization of the PAA representation of a time-series into SAX is implemented in a way which produces symbols corresponding to the time-series features with equal probability. The extensive and rigorous analysis of various time-series datasets available to the authors has shown that time-series that are normalized by the zero mean and unit of energy (2.2.4) follow the Normal distribution law. By using Gaussian distribution properties, it's easy to pick  $a$  equal-sized areas under the Normal curve using lookup tables [45] for the cut lines coordinates, slicing the under-the-Gaussian-curve area. The  $x$  coordinates of these lines called “breakpoints” in the SAX algorithm context. The list of breakpoints  $B = \beta_1, \beta_2, \dots, \beta_{a-1}$  such that  $\beta_{i-1} < \beta_i$  and  $\beta_0 = -\infty, \beta_a = \infty$  divides the area under  $N(0, 1)$  into  $a$  equal areas. By assigning a corresponding alphabet symbol  $alpha_j$  to each interval  $[\beta_{j-1}, \beta_j)$ , the conversion of the vector of PAA coefficients  $\bar{C}$  into the string  $\hat{C}$  implemented as follows:

$$\hat{c}_i = alpha_j, \text{ iif, } \bar{c}_i \in [\beta_{j-1}, \beta_j) \quad (3.6.0.20)$$

SAX introduces new metrics for measuring distance between strings by extending Euclidean (2.1.0.9) and PAA (3.5.0.19) distances. The function returning the minimal distance between two string representations of original time series  $\hat{Q}$  and  $\hat{C}$  is defined as

$$MINDIST(\hat{Q}, \hat{C}) \equiv \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^w (dist(\hat{q}_i, \hat{c}_i))^2} \quad (3.6.0.21)$$

where the  $dist$  function is implemented by using the lookup table for the particular set of the breakpoints (alphabet size) as shown in Table 3.1, and where the singular value for each cell  $(r, c)$

	a	b	c	d
a	0	0	0.67	1.34
b	0	0	0	0.67
c	0.67	0	0	0
d	1.34	0.67	0	0

Table 3.1: A lookup table used by the MINDIST function for the  $a = 4$

is computed as

$$cell_{(r,c)} = \begin{cases} 0, & \text{if } |r - c| \leq 1 \\ \beta_{\max(r,c)-1} - \beta_{\min(r,c)-1}, & \text{otherwise} \end{cases} \quad (3.6.0.22)$$

As shown by Li et al, this SAX distance metrics lower-bounds the PAA distance, i.e.

$$\sum_{i=1}^n (q_i - c_i)^2 \geq n(\bar{Q} - \bar{C})^2 \geq n(dist(\hat{Q}, \hat{C}))^2 \quad (3.6.0.23)$$

The SAX lower bound was examined by Ding et al [22] in great detail and found to be superior in precision to the spectral decomposition methods on bursty (non-periodic) data sets.

## Chapter 4

# Conclusion

There has been an increasing amount of research over the last two decades into time-series data mining, knowledge discovery and applications. It is very interesting to see how the focus of this research shifted from computationally intensive and mathematically elegant methods of spectral decomposition of time-series to extremely simple approximation methods such as PAA and SAX. In my opinion, this is due to the branching of the time-series data-mining field away from the classical time-series analysis school. While researchers started with adoption of the classical spectral analysis approach, which is almost perfect for resolving the “secret” of a time-series generative process, the time-series data-mining community, interested in the efficient navigation through the thousands of signals and petabytes of data, found its own extremely efficient ways to perform on-the-fly extraction and analysis of patterns. The latest application of SAX approximation to the image recognition [65] and monitoring insects real-time [36] by Keogh et al demonstrate how fast, accurate, and resource-efficient these techniques are.

# Bibliography

- [1] DTW-based motion comparison and retrieval. pages 211–226. 2007.
- [2] Dynamic Time Warping. pages 69–84. 2007.
- [3] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. pages 69–84. 1993.
- [4] Rakesh Agrawal, King-Ip Lin, Harpreet S. Sawhney, and Kyuseok Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *In VLDB*, pages 490–501, 1995.
- [5] T. W. Anderson. *The Statistical Analysis of Time Series*. Wiley-Interscience, first edition, June 1994.
- [6] J. Assfalg, H. P. Kriegel, P. Kroger, P. Kunath, A. Pryakhin, and M. Renz. T-time: Threshold-based data mining on time series. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 1620–1623, 2008.
- [7] Claus Bahlmann and Hans Burkhardt. The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(3):299–310, 2004.
- [8] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R\*-tree: an efficient and robust access method for points and rectangles. In *SIGMOD '90: Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, volume 19, pages 322–331, New York, NY, USA, June 1990. ACM Press.
- [9] R. Bellman and R. Kalaba. On adaptive control processes. *Automatic Control, IRE Transactions on*, 4(2):1–9, 1959.
- [10] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The X-tree: An index structure for high-dimensional data. In *VLDB '96: Proceedings of the 22th International Conference on Very Large Data Bases*, pages 28–39, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

- [11] Peter J. Brockwell and Richard A. Davis. *Time Series: Theory and Methods (Springer Series in Statistics)*. Springer, September 1998.
- [12] Yuhan Cai and Raymond Ng. Indexing spatio-temporal trajectories with chebyshev polynomials. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 599–610, New York, NY, USA, 2004. ACM.
- [13] K. Chakrabarti and S. Mehrotra. The hybrid tree: an index structure for high dimensional feature spaces. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 440–447, 1999.
- [14] Kaushik Chakrabarti, Eamonn Keogh, Sharad Mehrotra, and Michael Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27(2):188–228, June 2002.
- [15] Kin P. Chan and Wai C. Fu. Efficient time series matching by wavelets. *Data Engineering, International Conference on*, 0:126+, 1999.
- [16] Qiuxia Chen, Lei Chen, Xiang Lian, Yunhao Liu, and Jeffrey X. Yu. Indexable PLA for efficient similarity search. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 435–446. VLDB Endowment, 2007.
- [17] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.
- [18] Andrea Corradini. Dynamic time warping for off-line recognition of a small gesture vocabulary. In *RATFG-RTS '01: Proceedings of the IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (RATFG-RTS'01)*, Washington, DC, USA, 2001. IEEE Computer Society.
- [19] Jonathan D. Cryer and Kung-Sik Chan. *Time Series Analysis: With Applications in R (Springer Texts in Statistics)*. Springer, 2nd edition, October 2008.
- [20] Ingrid Daubechies. *Ten Lectures on Wavelets (C B M S - N S F Regional Conference Series in Applied Mathematics)*. Soc for Industrial & Applied Math, December 1992.
- [21] Denny and Vincent C. Lee. An alternative methodology for mining seasonal pattern using self-organizing map. pages 424–430. 2004.
- [22] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proc. VLDB Endow.*, 1(2):1542–1552, 2008.

- [23] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000.
- [24] Alon Efrat, Quanfu Fan, and Suresh Venkatasubramanian. Curve matching, time warping, and light fields: New algorithms for computing similarity between curves. *J. Math. Imaging Vis.*, 27(3):203–216, April 2007.
- [25] Waiyawuth Euachongprasit and Chotirat Ratanamahatana. Accurate and efficient retrieval of multimedia time series data under uniform scaling and time warping. pages 100–111. 2008.
- [26] Waiyawuth Euachongprasit and Chotirat Ratanamahatana. Efficient multimedia time series data retrieval under uniform scaling and normalisation. pages 506–513. 2008.
- [27] C. Faloutsos, H. V. Jagadish, A. O. Mendelzon, and T. Milo. A signature technique for similarity-based queries. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 2–20, 1997.
- [28] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD '94: Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, pages 419–429, New York, NY, USA, 1994. ACM Press.
- [29] Dina Goldin and Paris Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. pages 137–153. 1995.
- [30] Jie Gu and Xiaomin Jin. A simple approximation for dynamic time warping search in large time series database. pages 841–848. 2006.
- [31] A. Heck. Astronomical Time Series, Proceedings of The Florence and George Wise Observatory 25th Anniversary Symposium. *Space Science Reviews*, 85(3):549, August 1998.
- [32] Joseph M. Hellerstein, Elias Koutsoupias, and Christos H. Papadimitriou. On the analysis of indexing schemes. In *PODS '97: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 249–256, New York, NY, USA, 1997. ACM.
- [33] F. Itakura. Minimum prediction residual principle applied to speech recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 23(1):67–72, 1975.
- [34] T. Kahveci and A. Singh. Variable length queries for time series data. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 273–282, 2001.
- [35] T. Kahveci, A. Singh, and A. Gurel. Similarity searching for multi-attribute sequences. In *Scientific and Statistical Database Management, 2002. Proceedings. 14th International Conference on*, pages 175–184, 2002.



- [36] Shashwati Kasetty, Candice Stafford, Gregory P. Walker, Xiaoyue Wang, and Eamonn Keogh. Real-time classification of streaming sensor data. In *ICTAI '08: Proceedings of the 2008 20th IEEE International Conference on Tools with Artificial Intelligence*, pages 149–156, Washington, DC, USA, 2008. IEEE Computer Society.
- [37] Kelvin and Man H. Wong. Fast time-series searching with scaling and shifting. In *PODS '99: Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 237–248, New York, NY, USA, 1999. ACM.
- [38] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.
- [39] Eamonn Keogh, Jessica Lin, and Wagner Truppel. Clustering of time series subsequences is meaningless: Implications for previous and future research. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, Washington, DC, USA, 2003. IEEE Computer Society.
- [40] Eamonn J. Keogh and Michael J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback, 1998.
- [41] Flip Korn, H. V. Jagadish, and Christos Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 289–300, New York, NY, USA, 1997. ACM.
- [42] Flip Korn, Nikolaos Sidiropoulos, Christos Faloutsos, Eliot Siegel, and Zenon Protopapas. Fast nearest neighbor search in medical image databases. In *In Proceedings of the Int. Conf. on Very Large Data Bases*, pages 215–226, 1996.
- [43] Balaji Krishnapuram, Christopher M. Bishop, and Martin Szummer. Generative Models and Bayesian Model comparison for shape recognition. In *IWFHR '04: Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition*, pages 20–25, Washington, DC, USA, 2004. IEEE Computer Society.
- [44] A. Kuzmanic and V. Zanchi. Hand shape classification using DTW and LCSS as similarity measures for vision-based gesture recognition system. In *EUROCON, 2007. The International Conference on "Computer as a Tool"*, pages 264–269, 2007.
- [45] Richard J. Larsen and Morris L. Marx. *An Introduction to Mathematical Statistics and Its Applications (3rd Edition)*. Prentice Hall, 3rd edition, January 2000.

- [46] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *DMKD '03: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11, New York, NY, USA, 2003. ACM Press.
- [47] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, October 2007.
- [48] Jessica Lin, Eamonn J. Keogh, Stefano Lonardi, and Pranav Patel. Finding motifs in time series. In *2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, July 2002.
- [49] R. Martens and L. Claesen. On-line signature verification by dynamic time-warping. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, pages 38–42 vol.3, 1996.
- [50] L. Micó. Comparison of fast nearest neighbour classifiers for handwritten character recognition. *Pattern Recognition Letters*, 19(3-4):351–356, March 1998.
- [51] Y. Morinaka, M. Yoshikawa, T. Amagasa, and S. Uemura. The L-index: An indexing structure for efficient subsequence matching in timesequence databases. 2001.
- [52] M. Muller, H. Mattes, and F. Kurth. An efficient multiscale approach to audio synchronization. pages 192–197, 2006.
- [53] C. Myers, L. Rabiner, and A. Rosenberg. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *Acoustics, Speech, and Signal Processing [see also IEEE Transactions on Signal Processing]*, *IEEE Transactions on*, 28(6):623–635, 1980.
- [54] C. S. Myers. A comparative study of several dynamic time warping algorithms for speech recognition. Master’s thesis.
- [55] Richi Nayak and Paul T. Braak. Temporal pattern matching for the prediction of stock prices. In *AIDM '07: Proceedings of the 2nd international workshop on Integrating artificial intelligence and data mining*, pages 95–103, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.
- [56] V. Niennattrakul and C. A. Ratanamahatana. On clustering multimedia time series data using K-means and Dynamic Time Warping. In *Multimedia and Ubiquitous Engineering, 2007. MUE '07. International Conference on*, pages 733–738, 2007.

- [57] I. Popivanov and R. J. Miller. Similarity search over time-series data using wavelets. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 212–221, 2002.
- [58] Fatih Porikli and Tetsuji Haga. Event detection by eigenvector decomposition using object and frame features. *Computer Vision and Pattern Recognition Workshop*, 7:114+, 2004.
- [59] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43–49, 1978.
- [60] C. C. Tappert, C. Y. Suen, and T. Wakahara. The state of the art in online handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(8):787–808, 1990.
- [61] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press.
- [62] J. Vial, H. Nocairi, P. Sassiat, S. Mallipatu, G. Cognon, D. Thiebaut, B. Teillet, and D. Rutledge. Combination of dynamic time warping and multivariate analysis for the comparison of comprehensive two-dimensional gas chromatograms application to plant extracts. *Journal of Chromatography A*, September 2008.
- [63] Enrique Vidal and Francisco Casacuberta. On the verification of triangle inequality by dynamic time-warping dissimilarity measures. *Speech Commun.*, 7(1):67–79, 1988.
- [64] M. Vlachos, D. Gunopulos, and G. Kollios. Robust similarity measures for mobile object trajectories. In *Database and Expert Systems Applications, 2002. Proceedings. 13th International Workshop on*, pages 721–726, 2002.
- [65] Li Wei, Eamonn J. Keogh, and Xiaopeng Xi. Saxually explicit images: Finding unusual shapes. In *ICDM*, pages 711–720. IEEE Computer Society, 2006.
- [66] Nasser Yazdani and Meral Z. Özsoyoglu. Sequence matching of images. In *SSDBM '96: Proceedings of the Eighth International Conference on Scientific and Statistical Database Management*, pages 53–62, Washington, DC, USA, 1996. IEEE Computer Society.
- [67] Byoung-Kee Yi and Christos Faloutsos. Fast time sequence indexing for arbitrary Lp norms. In *VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases*, pages 385–394, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.
- [68] Jae Yoo, Myung Shin, Seok Lee, Kil Choi, Ki Cho, and Dae Hur. An efficient index structure for high dimensional image data. pages 131–144. 1999.
- [69] Zhang Zhang, Kaiqi Huang, and Tieniu Tan. Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes. In *ICPR '06: Proceedings of the 18th International*

*Conference on Pattern Recognition (ICPR'06)*, pages 1135–1138, Washington, DC, USA, 2006. IEEE Computer Society.

- [70] Yunyue Zhu and Dennis Shasha. Statstream: statistical monitoring of thousands of data streams in real time. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 358–369. VLDB Endowment, 2002.
- [71] Yunyue Zhu and Dennis Shasha. Warping indexes with envelope transforms for query by humming. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 181–192, New York, NY, USA, 2003. ACM.