Indexing of Compressed Time Series

Eugene Fink
Computer Science and Engineering
University of South Florida
Tampa, Florida 33620
eugene@csee.usf.edu

Kevin B. Pratt
Harris Corporation
1025 NASA Blvd., W3 / 9708
Melbourne, Florida 32919
kpratt01@harris.com

Abstract. We describe a procedure for identifying major minima and maxima of a time series, and present two applications of this procedure. The first application is fast compression of a series, by selecting major extrema and discarding the other points. The compression algorithm runs in linear time and takes constant memory. The second application is indexing of compressed series by their major extrema, and retrieval of series similar to a given pattern. The retrieval procedure searches for the series whose compressed representation is similar to the compressed pattern. It allows the user to control the trade-off between the speed and accuracy of retrieval. We show the effectiveness of the compression and retrieval for stock charts, meteorological data, and electroencephalograms.

Keywords. Time series, compression, fast retrieval, similarity measures.

1 Introduction

We view a *time series* as a sequence of values measured at equal intervals; for example, the series in Figure 1 includes the values 20, 22, 25, 22, and so on. We describe a compression procedure based on extraction of certain important minima and maxima from a series. For example, we can compress the series in Figure 1 by extracting the circled minima and maxima, and discarding the other points. We also propose a measure of similarity between series and show that it works well with compressed data. Finally, we present a technique for indexing and retrieval of compressed series; we have tested it on four data sets (Table 1), which are publicly available through the Internet.

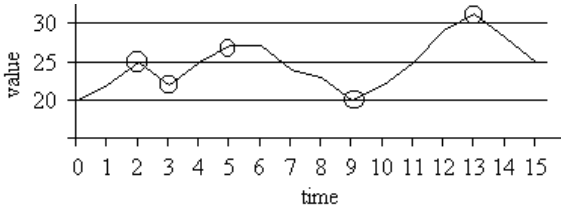


Figure 1 Example of a time series.

Table 1 Data sets used in the experiments.

data set	description	num. of series	points per series	total num. of points
stock prices	98 stocks, 2.3 years	98	610	60,000
air and sea temperatures	68 buoys, 18 years, 2 sensors per buoy	136	1,800–6,600	450,000
wind speeds	12 stations, 18 years	12	6,570	79,000
EEG	64 electrodes, 1 second	64	256	16,000

Stock prices: We have used stocks from the Standard and Poor's 100 listing of large companies for the period from January 1998 to April 2000. We have downloaded daily prices from America Online, discarded newly listed and de-listed stocks, and used ninety-eight stocks in the experiments.

Air and sea temperatures: We have experimented with daily temperature readings by sixty-eight buoys in the Pacific Ocean, from 1980 to 1998, downloaded from the Knowledge Discovery archive at the University of California at Irvine (kdd.ics.uci.edu).

Wind speeds: We have used daily wind speeds from twelve sites in Ireland, from 1961 to 1978, obtained from an archive at Carnegie Mellon University (lib.stat.cmu.edu/datasets).

Electroencephalograms: We have utilized EEG obtained by Henri Begleiter at the Neurodynamics Laboratory of the SUNY Health Center at Brooklyn. These data are from sixty-four electrodes at standard points on the scalp; we have downloaded them from an archive at the University of California at Irvine (kdd.ics.uci.edu).

2 Previous Work

We review related work on the comparison and indexing of time series.

Feature sets. Researchers have considered various feature sets for compressing time series and measuring similarity between them. They have

extensively studied Fourier transforms, which allow fast compression [Singh and McAtackney, 1998; Sheikholeslami *et al*., 1998; Stoffer, 1999; Yi *et al*., 2000]; however, this technique has several disadvantages. In particular, it smoothes local extrema, which may lead to a loss of important information, and it does not work well for erratic series [Ikeda *et al*., 1999]. Chan and his colleagues applied Haar wavelet transforms to time series and showed several advantages of this technique over Fourier transforms [Chan and Fu, 1999; Chan *et al*., 2003].

Guralnik and Srivastava [1999] considered the problem of detecting a change in the trend of a data stream, and developed a technique for finding "change points" in a series. Last *et al.* [2001] proposed a general framework for knowledge discovery in time series, which included representation of a series by its key features, such as slope and signal-to-noise ratio. They described a technique for computing these features and identifying the points of change in the feature values.

Researchers have also studied the use of small alphabets for compression of time series, and applied string matching to the pattern search [Agrawal *et al*., 1995; Huang and Yu, 1999; André-Jönsson and Badal, 1997; Lam and Wong, 1998; Park *et al*., 1999; Qu *et al*., 1998]. For example, Guralnik *et al.* [1997] compressed stock prices using a nine-letter alphabet. Singh and McAtackney [1998] represented stock prices, particle dynamics, and stellar light intensity using a three-letter alphabet. Lin and Risch [1998] used a two-letter alphabet to encode major spikes in a series. Das *et al.* [1998] utilized an alphabet of primitive shapes for efficient compression. These techniques give a high compression rate, but their descriptive power is limited, which makes them inapplicable in many domains.

Perng *et al.* [2000] investigated a compression technique based on extraction of "landmark points," which included local minima and maxima. Keogh and Pazzani [1997; 1998] used the endpoints of best-fit line segments to compress a series. Keogh *et al.* [2001] reviewed the compression techniques based on approximation of a time series by a sequence of straight segments. We describe an alternative compression technique, based on selection of important minima and maxima.

Similarity measures. Several researchers have defined similarity as the distance between points in a feature space. For example, Caraca-Valente and Lopez-Chavarrias [2000] used Euclidean distance between feature vectors containing angle of knee movement and muscle strength, and Lee

et al. [2000] applied Euclidean distance to compare feature vectors containing color, texture, and shape of video data. This technique works well when all features have the same units of scale [Goldin and Kanellakis, 1995], but it is often ineffective for combining disparate features.

An alternative definition of similarity is based on bounding rectangles; two series are similar if their bounding rectangles are similar. It allows fast pruning of clearly dissimilar series [Perng *et al*., 2000; Lee *et al*., 2000], but it is less effective for selecting the most similar series.

The envelope-count technique is based on dividing a series into short segments, called envelopes, and defining a yes/no similarity for each envelope. Two series are similar within an envelope if their point-by-point differences are within a certain threshold. The overall similarity is measured by the number of envelopes where the series are similar [Agrawal *et al*., 1996]. This measure allows fast computation of similarity, and it can be adapted for noisy and missing data [Das *et al*., 1997; Bollobas *et al*., 1997].

Finally, we can measure a point-by-point similarity of two series and then aggregate these measures, which often requires interpolation of missing points. For example, Keogh and Pazzani [1998] used linear interpolation with this technique, and Perng *et al.* [2000] applied cubic approximation. Keogh and Pazzani [2000] also described a point-by-point similarity with modified Euclidean distance, which does not require interpolation.

Indexing and retrieval. Researchers have studied a variety of techniques for indexing of time series. For example, Deng [1998] applied *kd*-trees to arrange series by their significant features, Chan and Fu [1999] combined wavelet transforms with R-trees, and Bozkaya and her colleagues used vantage-point trees for indexing series by numeric features [Bozkaya *et al*., 1997; Bozkaya and Özsoyoglu, 1999]. Park *et al.* [2001] indexed series by their local extrema and by properties of the segments between consecutive extrema. Li *et al.* [1998] proposed a retrieval technique based on a multi-level abstraction hierarchy of features. Aggarwal and Yu [2000] considered grid structures, but found that the grid performance is often no better than exhaustive search. They also showed that exhaustive search among compressed series if often faster than sophisticated indexing techniques.

3 Important Points

We compress a time series by selecting some of its minima and maxima, and dropping the other points (Figure 2). The intuitive idea is to discard minor fluctuations and keep major minima and maxima. We control the compression rate with a parameter R, which is always greater than one; an increase of R leads to selection of fewer points. A point a_m of a series $a_1,...,a_n$ is an *important minimum* if there are indices i and j, where $i \le m \le j$, such that

- a_m is the minimum among $a_i,...,a_j$, and
- $a_i / a_m \ge R$ and $a_j / a_m \ge R$.

Intuitively, a_m is the minimal value of some segment $a_i,...,a_j$, and the endpoint values of this segment are much larger than a_m (Figure 3). Similarly, a_m is an *important maximum* if there are indices i and j, where $i \le m \le j$, such that

- a_m is the maximum among $a_i,...,a_j$, and
- $a_m / a_i \ge R$ and $a_m / a_j \ge R$.

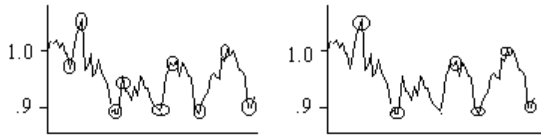


Figure 2 Important points for 91% compression (left) and 94% compression (right).

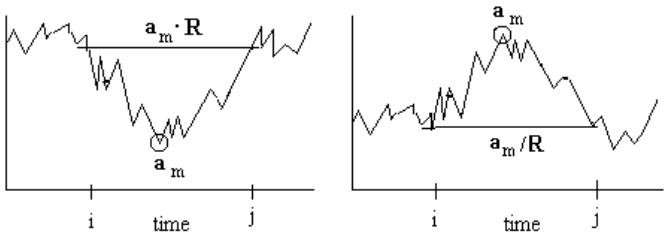


Figure 3 Important minimum (left) and important maximum (right).

In Figure 4, we give a procedure for selecting important points, which takes linear time and constant memory. It outputs the values and indices of all important points, as well as the first and last point of the series. This procedure can process new points as they arrive, without storing the original series; for example, it can compress a live electroencephalogram without waiting until the end of the data collection. We

have implemented it in Visual Basic 6 and tested on a 300-MHz PC; for an n-point series, the compression time is $14 \cdot n$ microseconds.

We have applied the compression procedure to the data sets in Table 1, and compared it with two simple techniques: equally spaced points and randomly selected points. We have experimented with different *compression rates,* which are defined as the percentage of points removed from a series. For example, "eighty-percent compression" means that we select 20% of points and discard the other 80%.

For each compression technique, we have measured the difference between the original series and the compressed series. We have considered three measures of difference between the original series, $a_1,...,a_n$, and the series interpolated from the compressed version, $b_1,...,b_n$.

Mean difference: $\frac{1}{n} \cdot \sum_{i=1}^{n} |a_i - b_i|$.

Maximum difference: $\max_{i \in [1..n]} |a_i - b_i|$.

Root mean square difference: $\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} (a_i - b_i)^2}$.

We summarize the results in Table 2, which shows that important points are significantly more accurate than the two simple methods.

4 Similarity Measures

We define similarity between time series, which underlies the retrieval procedure. We measure similarity on a zero-to-one scale; zero means no likeness and one means perfect likeness. We review three basic measures of similarity and then propose a new measure. First, we define similarity between two numbers, a and b:

$$\mathrm{sim}(a,b) = 1 - \frac{|a-b|}{|a|+|b|}.$$

The *mean similarity* between two series, $a_1,...,a_n$ and $b_1,...,b_n$, is the mean of their point-by-point similarity:

$$\frac{1}{n} \cdot \sum_{i=1}^{n} \mathrm{sim}(a_i, b_i).$$

We also define the *root mean square similarity:*

$$\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} \mathrm{sim}(a_i, b_i)^2}.$$

IMPORTANT-POINTS — Top-level function for finding important points.
The input is a time series $a_1,...,a_n$; the output is the
values and indices of the selected important points.

output $(a_1, 1)$
i = FIND-FIRST
if $i < n$ and $a_i > a_1$ **then** i = FIND-MAXIMUM(i)
while $i < n$ **do**
    i = FIND-MINIMUM(i)
    i = FIND-MAXIMUM(i)
output (a_n, n)

FIND-FIRST — Find the first important point.
$iMin = 1$; $iMax = 1$; $i = 2$
while $i < n$ and $a_i / a_{iMin} < R$ and $a_{iMax} / a_i < R$ **do**
    if $a_i < a_{iMin}$ **then** $iMin = i$
    if $a_i > a_{iMax}$ **then** $iMax = i$
    $i = i + 1$
if $iMin < iMax$
    then output $(a_{iMin}, iMin)$
    else output $(a_{iMax}, iMax)$
return i

FIND-MINIMUM(i) — Find the first important minimum after the ith point.
$iMin = i$
while $i < n$ and $a_i / a_{iMin} < R$ **do**
    if $a_i < a_{iMin}$ **then** $iMin = i$
    $i = i + 1$
if $i < n$ or $a_{iMin} < a_i$ **then output** $(a_{iMin}, iMin)$
return i

FIND-MAXIMUM(i) — Find the first important maximum after the ith point.
$iMax = i$
while $i < n$ and $a_{iMax} / a_i < R$ **do**
    if $a_i > a_{iMax}$ **then** $iMax = i$
    $i = i + 1$
if $i < n$ or $a_{iMax} > a_i$ **then output** $(a_{iMax}, iMax)$
return i

Figure 4 Compression procedure. We process a series $a_1,...,a_n$ and use a global variable n to represent its size. The procedure outputs the values and indices of the selected points.

Table 2 Accuracy of three compression techniques. We give the average difference between an original series and its compressed version using the three difference measures; smaller differences correspond to more accurate compression.

	mean difference			**maximum difference**			**root mean square diff.**		
	important points	fixed points	random points	important points	fixed points	random points	important points	fixed points	random points
eighty percent compression									
stock prices	0.02	0.03	0.04	0.70	1.70	1.60	0.05	0.14	0.14
air temp.	0.01	0.03	0.03	0.33	0.77	0.72	0.03	0.10	0.10
sea temp.	0.01	0.03	0.03	0.35	0.81	0.75	0.03	0.10	0.10
wind speeds	0.02	0.03	0.03	0.04	1.09	1.01	0.04	0.05	0.05
EEG	0.03	0.06	0.07	0.68	1.08	1.00	0.10	0.18	0.17
ninety percent compression									
stock prices	0.03	0.06	0.07	1.10	1.70	1.70	0.08	0.21	0.21
air temp.	0.02	0.05	0.05	0.64	0.80	0.78	0.08	0.16	0.14
sea temp.	0.01	0.04	0.05	0.60	0.83	0.82	0.07	0.16	0.14
wind speeds	0.03	0.04	0.04	0.06	1.09	1.03	0.05	0.06	0.06
EEG	0.08	0.13	0.12	0.82	1.10	1.09	0.17	0.27	0.24
ninety-five percent compression									
stock prices	0.05	0.10	0.12	1.30	1.80	1.80	0.11	0.32	0.30
air temp.	0.03	0.09	0.08	0.74	0.83	0.83	0.12	0.23	0.21
sea temp.	0.03	0.08	0.08	0.78	0.85	0.85	0.12	0.23	0.21
wind speeds	0.05	0.04	0.04	0.08	1.09	1.10	0.07	0.08	0.08
EEG	0.13	0.17	0.16	0.90	1.10	1.10	0.24	0.31	0.28

In addition, we consider the correlation coefficient, which is a standard statistical measure of similarity. It ranges from −1 to 1, but we can convert it to the zero-to-one scale by adding one and dividing by two. For series $a_1,...,a_n$ and $b_1,...,b_n$, with mean values $m_a = (a_1 +...+ a_n) / n$ and $m_b = (b_1 +...+ b_n) / n$, the correlation coefficient is

$$\frac{\sum_{i=1}^{n} (a_i - m_a) \cdot (b_i - m_b)}{\sqrt{\sum_{i=1}^{n} (a_i - m_a)^2 \cdot \sum_{i=1}^{n} (b_i - m_b)^2}}.$$

We next define a new similarity measure, called the *peak similarity.* For numbers a and b, their peak similarity is

$$\mathrm{psim}(a,b) = 1 - \frac{|a - b|}{2 \cdot \max(|a|,|b|)}.$$

In Figure 5, we show the meaning of this definition for $|a| \geq |b|$, based on the illustrated triangle. We draw the vertical line through b to the in-

tersection with the triangle's side; the ordinate of the intersection is the similarity of a and b. The peak similarity of two series is the mean similarity of their points, $(\text{psim}(a_1, b_1) + ... + \text{psim}(a_n, b_n)) / n$.

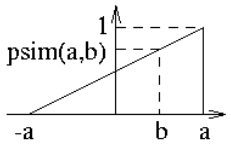


Figure 5 Peak similarity of numbers a and b, where $|a| \geq |b|$.

We next give an empirical comparison of the four similarity measures. For each series, we have found the five most similar series, and then determined the mean difference between the given series and the other five. In Table 3, we summarize the results and compare them with the perfect exhaustive-search selection and with random selection. The results show that the peak similarity performs better than the other measures, and that the correlation coefficient is the least effective.

We have also used the four similarity measures to identify close matches for each series, and compared the results with ground-truth neighborhoods. For stocks, we have considered small neighborhoods formed by industry subgroups, as well as large neighborhoods formed by industry groups, according to Standard and Poor's classification. For air and sea temperatures, we have used geographic proximity to define two ground-truth neighborhoods. The first neighborhood is the 1×5 rectangle in the grid of buoys, and the second is the 3×5 rectangle. For wind speeds, we have also used geographic proximity; the first neighborhood includes all sites within 70 miles, and the second includes the sites within 140 miles. For electroencephalograms, the first neighborhood is the 3×3 rectangle in the grid of electrodes, and the second is the 5×5 rectangle.

For each series, we have found the five closest matches, and then determined the average number of matches that belong to the same neighborhood. In Table 4, we give the results and compare them with the perfect selection and random selection; larger numbers correspond to better selections.

The results show that the peak similarity is usually more effective than the other three similarities. If we use the 90% compression, the peak similarity gives better selection than the other similarity measures for the stock prices, air and sea temperatures, and EEG; however, it gives worse results for the wind speeds. If we use the 95% compression, the peak

similarity outperforms the other measures on the stocks, temperatures, EEG, and large-neighborhood selection for the wind speeds; however, it loses to the mean similarity and correlation coefficient on the small-neighborhood selection for the wind speeds.

We have also checked how well the peak similarity of original series correlates with the peak similarity of their compressed versions (Figure 6). We have observed a good linear correlation, which gracefully degrades with an increase of compression rate.

Table 3 Differences between selected similar series. For each given series, we have selected the five most similar series, and then measured the mean difference between the given series and the other five. Smaller differences correspond to better selection. We also show the running times of selecting similar series.

similarity measure	comp. rate	**stock prices** mean diff.	max diff.	time (sec)	**sea temperatures** mean diff.	max diff.	time (sec)	**air temperatures** mean diff.	max diff.	time (sec)
perfect selection		0.094	0.437		0.016	0.072		0.024	0.121	
random selection		0.287	1.453		0.078	0.215		0.070	0.235	
peak similarity	90%	0.103	0.429	0.024	0.018	0.068	0.021	0.029	0.103	0.022
	95%	0.110	0.534	0.022	0.019	0.073	0.019	0.030	0.136	0.020
mean similarity	90%	0.110	0.525	0.026	0.026	0.092	0.022	0.031	0.134	0.022
	95%	0.126	0.570	0.024	0.033	0.112	0.021	0.037	0.152	0.022
root mean square sim.	90%	0.103	0.497	0.026	0.024	0.090	0.022	0.030	0.133	0.022
	95%	0.115	0.588	0.024	0.031	0.106	0.021	0.035	0.147	0.022
correlation coefficient	90%	0.206	1.019	0.048	0.054	0.162	0.044	0.051	0.214	0.046
	95%	0.210	1.101	0.045	0.063	0.179	0.042	0.051	0.224	0.043

similarity measure	comp. rate	**wind speeds** mean diff.	max diff.	time (sec)	**EEG** mean diff.	max diff.	time (sec)
perfect selection		0.021	0.136		0.038	0.170	
random selection		0.029	0.185		0.072	0.370	
peak similarity	90%	0.023	0.138	0.016	0.052	0.241	0.015
	95%	0.023	0.148	0.016	0.063	0.306	0.015
mean similarity	90%	0.023	0.137	0.017	0.055	0.279	0.016
	95%	0.025	0.152	0.017	0.066	0.323	0.014
root mean square sim.	90%	0.023	0.134	0.017	0.051	0.261	0.016
	95%	0.023	0.153	0.017	0.064	0.317	0.014
correlation coefficient	90%	0.024	0.138	0.042	0.056	0.281	0.030
	95%	0.024	0.154	0.033	0.068	0.349	0.028

Table 4 Finding members of the same neighborhood. For each series, we have found the five closest matches, and then determined the average number of the series among them that belong to the same ground-truth neighborhood.

similarity measure	comp. rate	**stocks** 1	2	**sea temp.** 1	2	**air temp.** 1	2	**wind speeds** 1	2	**EEG** 1	2
perfect selection		5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
random selection		0.07	0.29	0.11	0.40	0.10	0.40	0.74	2.27	0.35	1.03
peak similarity	90%	0.22	0.62	0.54	1.09	0.49	0.89	1.16	2.83	1.81	2.81
	95%	0.21	0.55	0.65	1.18	0.48	0.82	1.50	2.83	1.59	2.25
mean similarity	90%	0.18	0.55	0.28	0.85	0.34	0.77	1.33	2.92	1.05	1.98
	95%	0.12	0.47	0.17	0.75	0.25	0.65	1.58	2.66	0.36	0.90
root mean square sim.	90%	0.14	0.53	0.32	0.88	0.34	0.83	1.50	2.92	1.20	2.19
	95%	0.17	0.35	0.20	0.77	0.26	0.71	1.33	2.75	0.36	0.90
correlation coefficient	90%	0.15	0.39	0.25	0.82	0.49	0.74	1.33	2.92	1.16	2.24
	95%	0.19	0.50	0.29	0.72	0.34	0.60	1.51	2.75	0.68	1.65

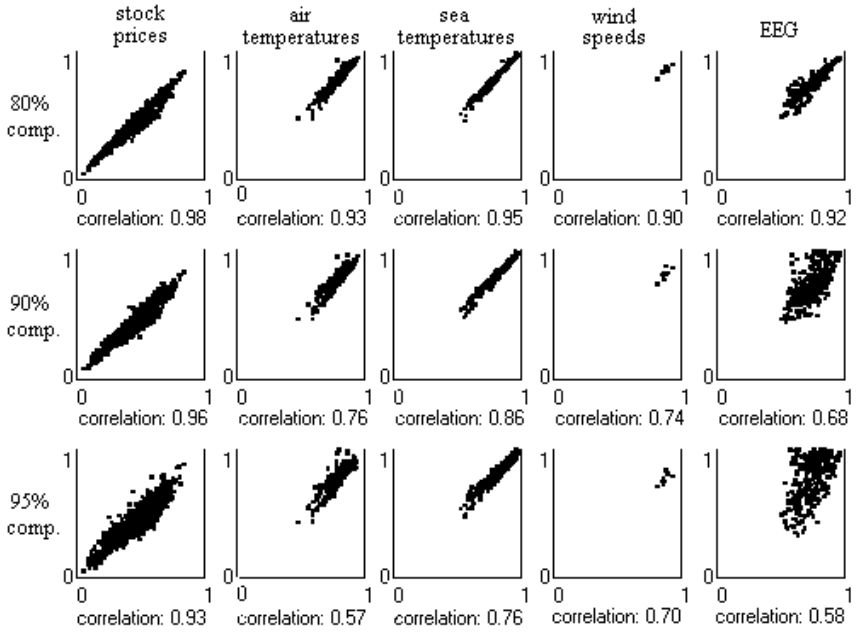


Figure 6 Correlation between the peak similarity of original series and the peak similarity of their compressed versions. We show the correlation for three compression rates: 80%, 90%, and 95%.

5 Pattern Retrieval

We give an algorithm that inputs a pattern series and retrieves similar series from a database. It includes three steps: identifying a "prominent feature" of the pattern, finding similar features in the database, and comparing the pattern with each series that has a similar feature.

We begin by defining a *leg* of a series, which is the segment between two consecutive important points. For each leg, we store the values listed in Figure 7, denoted *vl*, *vr*, *il*, *ir*, *ratio*, and *length*; we give an example of these values in Figure 8. The *prominent leg* of a pattern is the leg with the greatest endpoint ratio.

The retrieval procedure inputs a pattern and searches for similar segments in a database (Figure 9). First, it finds the pattern leg with the greatest endpoint ratio, denoted $ratio_p$, and determines the length of this leg, $length_p$. Next, it identifies all legs in the database that have a similar endpoint ratio and length. A leg is considered similar to the pattern leg if its ratio is between $ratio_p / C$ and $ratio_p \cdot C$, and its length is between $length_p / D$ and $length_p \cdot D$, where C and D are parameters for controlling the search.

We index all legs in the database by their ratio and length using a *range tree,* which is a standard structure for indexing points by two coordinates [Edelsbrunner, 1981; Samet, 1990]. If the total number of legs is l, and the number of retrieved legs with an appropriate ratio and length is k, then the retrieval time is $O(k + \lg l)$.

Finally, the procedure identifies the segments that contain the selected legs (Figure 10) and computes their similarity to the pattern. If the similarity is above a given threshold T, the procedure outputs the segment as a match. In Figure 11, we give an example of a stock-price pattern and similar segments retrieved from the stock database.

The described procedure can miss a matching segment that does not have a leg corresponding to the pattern's prominent leg. We illustrate this problem in Figure 12, where the prominent leg of the pattern has no equivalent in the matching series. To avoid this problem, we introduce the notion of an *extended leg,* which is a segment that would be a leg under a higher compression rate (Figure 12c). Formally, points a_i and a_j of a series $a_1,...,a_n$ form an extended upward leg if

- a_i is a local minimum, and a_j is a local maximum, and
- for every $m \in [i..j]$, we have $a_i < a_m < a_j$.

The definition of an extended downward leg is similar.

vl	value of the left important point of the leg
vr	value of the right important point of the leg
il	index of the left important point
ir	index of the right important point
ratio	ratio of the endpoints, defined as *vr* / *vl*
length	length of the leg, defined as *ir* − *il*

Figure 7 Basic data for a leg.

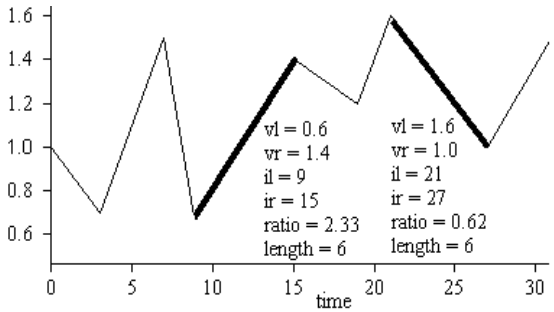


Figure 8 Example legs. We show the basic data for the two legs marked by thick lines.

PATTERN-RETRIEVAL
The procedure inputs a pattern series and searches a time-series database;
the output is a list of segments from the database that match the pattern.

Identify the pattern leg p with the greatest endpoint ratio, denoted $ratio_p$.
Determine the length of this pattern leg, denoted $length_p$.
Find all legs in the database that satisfy the following conditions:

- their endpoint ratios are between $ratio_p / C$ and $ratio_p \cdot C$, and
- their lengths are between $length_p / D$ and $length_p \cdot D$.

For each leg in the set of selected legs:
Identify the segment corresponding to the pattern (Figure 10).
Compute the similarity between the segment and the pattern.
If the similarity is above the threshold T, then output the segment.

Figure 9 Search for segments similar to a given pattern. We use three parameters to control the search: maximal ratio deviation C, maximal length deviation D, and similarity threshold T.

Figure 10 Identifying a segment that may match the pattern.

Figure 11 Example of retrieved stock charts.

Figure 12 Example of extended legs. The pattern (a) matches the series (b), but the pattern's prominent leg has no equivalent in the series. If we identify the extended legs (c), the prominent leg matches one of them.

We identify all extended legs, and index them in the same way as normal legs. The advantage of this approach is more accurate retrieval, and the disadvantage is a larger indexing structure. In Figure 13, we give an algorithm for identifying upward extended legs; the procedure for finding downward extended legs is similar. We assume that normal upward legs in the input series are numbered from 1 to l. First, the procedure processes important maxima; for each maximum ir_k, it identifies the *next larger* maximum and stores its index in $next[k]$. Second, it uses this information to identify extended legs. The running time of the first part is linear in the number of normal legs, and the time of the second part is linear in the number of extended legs.

```
EXTENDED-LEGS
The input is the list of legs in a series;
the output is a list of all extended legs.

initialize an empty stack S of leg indices
PUSH(S, 1)
for k = 2 to l do
    while S is not empty and ir_TOP(S) < ir_k do
        next[TOP(S)] = k; POP(S)
    PUSH(k)
while S is not empty do
    next[TOP(S)] = NIL; POP(S)
initialize an empty list of extended legs
for k = 1 to l − 1 do
    m = next[k]
    while m is not NIL do
        add (il_k, ir_m) to the list of extended legs
        m = next[m]
```

Figure 13 Identifying extended legs. We assume that normal legs are numbered 1 to l.

To evaluate the retrieval accuracy, we have compared the retrieval results with the matches identified by a slow exhaustive search. We have ranked the matches found by the retrieval algorithm from most to least similar. In Figures 14 and 15, we plot the ranks of matches found by the fast algorithm versus the ranks of exhaustive-search matches. For instance, if the fast algorithm has found only three among seven closest matches, the graph includes the point (3, 7). Note that the fast algorithm never returns "false positives" since it verifies each candidate match.

The retrieval time grows linearly with the pattern length and with the number of candidate segments identified at the initial step of the retrieval algorithm. If we increase C and D, the procedure finds more candidates and misses fewer matchers, but the retrieval takes more time. In Table 5, we give the mean number of candidate segments in the retrieval experiments, for three different values of C and D; note that this number does not depend on the pattern length.

We have measured the speed of a Visual-Basic implementation on a 300-MHz PC. If the pattern has m legs, and the procedure identifies k candidate matches, then the retrieval time is $70 \cdot m \cdot k$ microseconds. For the stock database with 60,000 points, the retrieval takes from 0.1 to 2.5 seconds. For the database of air and sea temperatures, which includes 450,000 points, the time is between 1 and 10 seconds.

Table 5 Average number of the candidate segments that match a pattern's prominent leg. The retrieval algorithm identifies these candidates and then compares them with the pattern. The number of candidates depends on the search parameters *C* and *D*.

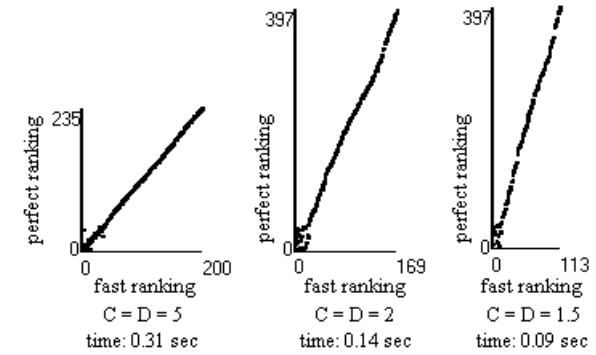
search parameters	**stock prices**	**air and sea temperatures**	**wind speeds**	**EEG**
$C = D = 1.5$	270	1,300	970	40
$C = D = 2$	440	2,590	1,680	70
$C = D = 5$	1,090	11,230	2,510	220

6 Concluding Remarks

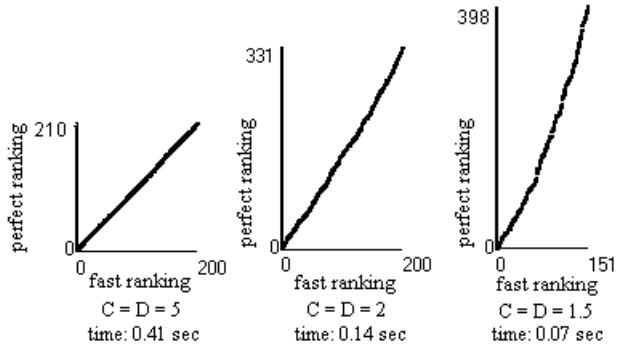
The main results include a procedure for compressing time series, indexing of compressed series by their prominent features, and retrieval of series whose compressed representation is similar to the compressed pattern. The experiments have shown the effectiveness of this technique for indexing of stock prices, weather data, and electroencephalograms. We plan to apply it to other time-series domains and study the factors that affect its effectiveness.

We are working on an extended version of the compression procedure, which will assign different importance levels to the extrema of time series, and allow construction of a hierarchical indexing structure [Gandhi, 2003]. We also aim to extend the developed technique for finding patterns that are stretched over time, and apply it to identifying periodic patterns, such as weather cycles.

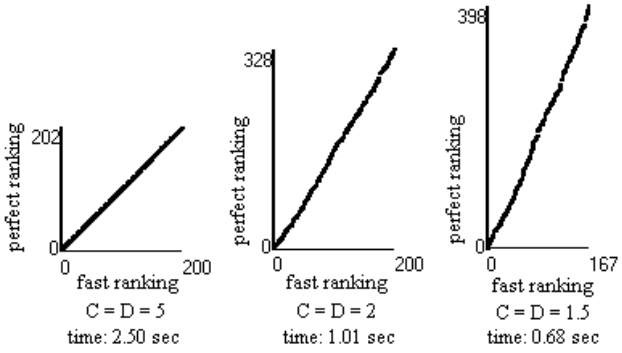
Acknowledgements. We are grateful to Mark Last, Eamonn Keogh, Dmitry Goldgof, and Rafael Perez for their valuable comments and suggestions. We also thank Savvas Nikiforou for his comments and help with software installations.



(a) Search for four-leg patterns.

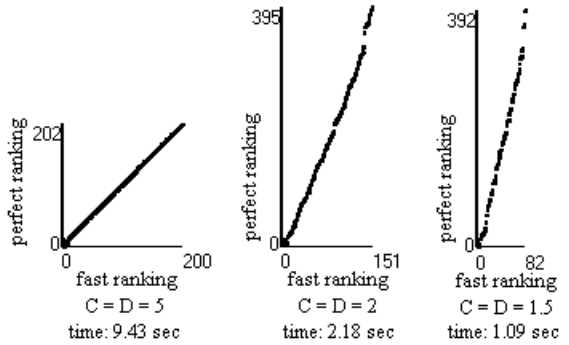


(b) Search for six-leg patterns.

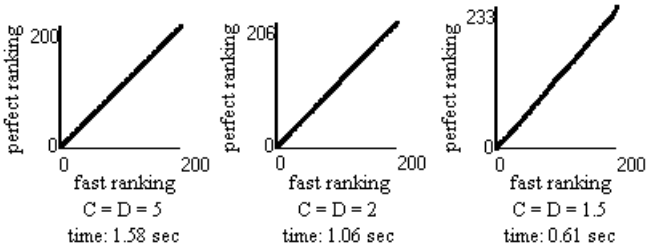


(c) Search for thirty-leg patterns.

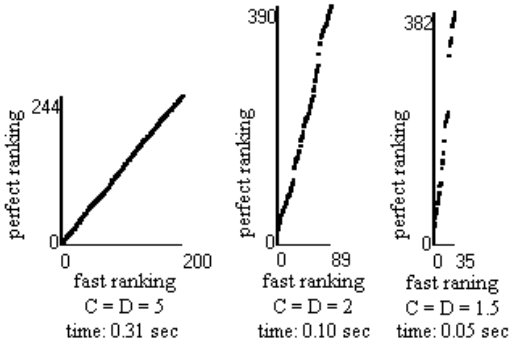
Figure 14 Retrieval of stock charts. The horizontal axes show the ranks of matches retrieved by the fast algorithm. The vertical axes are the ranks assigned to the same matches by the exhaustive search. If the fast algorithm has found all matches, the graph is a forty-five degree line; otherwise, it is steeper.



(a) Search for twelve-leg patterns of air and sea temperatures.



(b) Search for nine-leg patterns of wind speeds.



(c) Search for electroencephalogram patterns with twenty legs.

Figure 15 Retrieval of weather and electroencephalogram patterns. The horizontal axes show the similarity ranks assigned by the fast algorithm, and the vertical axes are the exhaustive-search ranks.

References

[Aggarwal and Yu, 2000] Charu C. Aggarwal and Philip S. Yu. The IGrid index: Reversing the dimensionality curse for similarity indexing in high-dimensional space. In *Proceedings of the Sixth ACM International Conference on Knowledge Discovery and Data Mining,* pages 119–129, 2000.

[Agrawal *et al*., 1995] Rakesh Agrawal, Giuseppe Psaila, Edward L. Wimmers, and Mohamed Zait. Querying shapes of histories. In *Proceedings of the Twenty-First International Conference on Very Large Data Bases,* pages 502–514, 1995.

[Agrawal *et al*., 1996] Rakesh Agrawal, Manish Mehta, John C. Shafer, Ramakrishnan Srikant, Andreas Arning, and Toni Bollinger. The Quest data mining system. In *Proceedings of the Second ACM International Conference on Knowledge Discovery and Data Mining,* pages 244–249, 1996.

[André-Jönsson and Badal, 1997] Henrik André-Jönsson and Dushan Z. Badal. Using signature files for querying time-series data. In *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery,* pages 211–220, 1997.

[Bollobas *et al*., 1997] Bela Bollobas, Gautam Das, Dimitrios Gunopulos, and Heikki Mannila. Time-series similarity problems and well-separated geometric sets. In *Proceedings of the Thirteenth Annual Symposium on Computational Geometry,* pages 454–456, 1997.

[Bozkaya and Özsoyoglu, 1999] Tolga Bozkaya and Z. Meral Özsoyoglu. Indexing large metric spaces for similarity search queries. *ACM Transactions on Database Systems,* 24(3):361–404, 1999.

[Bozkaya *et al*., 1997] Tolga Bozkaya, Nasser Yazdani, and Z. Meral Özsoyoglu. Matching and indexing sequences of different lengths. In *Proceedings of the Sixth International Conference on Information and Knowledge Management,* pages 128–135, 1997.

[Brockwell and Davis, 1996] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting.* Springer-Verlag, Berlin, Germany, 1996.

[Burrus *et al*., 1997] C. Sidney Burrus, Ramesh A. Gopinath, and Haitao Guo. *Introduction to Wavelets and Wavelet Transforms: A Primer.* Prentice Hall, Englewood Cliffs, NJ, 1997.

[Caraca-Valente and Lopez-Chavarrias, 2000] Juan Pedro Caraca-Valente and Ignacio Lopez-Chavarrias. Discovering similar patterns in time series. In *Proceedings of the Sixth ACM International Conference on Knowledge Discovery and Data Mining,* pages 497–505, 2000.

[Chan and Fu, 1999] Kin-Pong Chan and Ada Wai-Chee Fu. Efficient time series matching by wavelets. In *Proceedings of the Fifteenth International Conference on Data Engineering,* pages 126–133, 1999.

[Chan *et al*., 2003] Kin-Pong Chan, Ada Wai-Chee Fu, and C. Yu. Haar wavelets for efficient similarity search of time-series: With and without time warping. *IEEE Transactions on Knowledge and Data Engineering,* 2003. Forthcoming.

[Chi *et al*., 1995] Ed Huai-Hsin Chi, Phillip Barry, Elizabeth Shoop, John V. Carlis, Ernest Retzel, and John Riedl. Visualization of biological sequence similarity search results. In *Proceedings of the IEEE Conference on Visualization,* pages 44–51, 1995.

[Cortes *et al*., 2000] Corinna Cortes, Kathleen Fisher, Daryl Pregibon, and Anne Rogers. Hancock: A language for extracting signatures from data streams. In *Proceedings of the Sixth ACM International Conference on Knowledge Discovery and Data Mining,* pages 9–17, 2000.

[Das *et al*., 1997] Gautam Das, Dimitrios Gunopulos, and Heikki Mannila. Finding similar time series. In *Proceedings of the First European Conference on Principles of Data Mining and Knowledge Discovery,* pages 88–100, 1997.

[Das *et al*., 1998] Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule discovery from time series. In *Proceedings of the Fourth ACM International Conference on Knowledge Discovery and Data Mining,* pages 16–22, 1998.

[Deng, 1998] Kan Deng. *OMEGA: On-Line Memory-Based General Purpose System Classifier.* PhD thesis, Robotics Institute, Carnegie Mellon University, 1998. Technical Report CMU-RI-TR-98-33.

[Domingos and Hulten, 2000] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM International Conference on Knowledge Discovery and Data Mining,* pages 71–80, 2000.

[Edelsbrunner, 1981] Herbert Edelsbrunner. A note on dynamic range searching. *Bulletin of the European Association for Theoretical Computer Science,* 15:34–40, 1981.

[Fountain *et al*., 2000] Tony Fountain, Thomas G. Dietterich, and Bill Sudyka. Mining IC test data to optimize VLSI testing. In *Proceedings of the Sixth ACM International Conference on Knowledge Discovery and Data Mining,* pages 18–25, 2000.

[Franses, 1998] Philip Hans Franses. *Time Series Models for Business and Economic Forecasting.* Cambridge University Press, Cambridge, United Kingdom, 1998.

[Galka, 2000] Andreas Galka. *Topics in Nonlinear Time Series Analysis with Implications for EEG Analysis.* World Scientific, Singapore, 2000.

[Gandhi, 2003] Harith Suman Gandhi. Important extrema of time series: Theory and applications. Master's thesis, Computer Science and Engineering, University of South Florida, 2003. Forthcoming.

[Gavrilov *et al*., 2000] Martin Gavrilov, Dragomir Anguelov, Piotr Indyk, and Rajeev Motwani. Mining the stock market: Which measure is best. In *Proceedings of the Sixth ACM International Conference on Knowledge Discovery and Data Mining,* pages 487–496, 2000.

[Geva, 1999] Amir B. Geva. Hierarchical-fuzzy clustering of temporal patterns and its application for time-series prediction. *Pattern Recognition Letters,* 20(14):1519–1532, 1999.

[Goldin and Kanellakis, 1995] Dina Q. Goldin and Paris C. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In *Proceedings of the First International Conference on Principles and Practice of Constraint Programming,* pages 137–153, 1995.

[Graps, 1995] Amara L. Graps. An introduction to wavelets. *IEEE Computational Science and Engineering,* 2(2):50–61, 1995.

[Grenander, 1996] Ulf Grenander. *Elements of Pattern Theory.* Johns Hopkins University Press, Baltimore, MD, 1996.

[Guralnik and Srivastava, 1999] Valery Guralnik and Jaideep Srivastava. Event detection from time series data. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pages 33–42, 1999.

[Guralnik *et al*., 1998] Valery Guralnik, Duminda Wijesekera, and Jaideep Srivastava. Pattern-directed mining of sequence data. In *Proceedings of the Fourth ACM International Conference on Knowledge Discovery and Data Mining,* pages 51–57, 1998.

[Gusfield, 1997] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, Cambridge, United Kingdom, 1997.

[Han *et al*., 1998] Jiawei Han, Wan Gong, and Yiwen Yin. Mining segment-wise periodic patterns in time-related databases. In *Proceedings of the Fourth ACM International Conference on Knowledge Discovery and Data Mining,* pages 214–218, 1998.

[Haslett and Raftery, 1989] John Haslett and Adrian E. Raftery. Space-time modeling with long-memory dependence: Assessing Ireland's wind power resource. *Applied Statistics,* 38:1–50, 1989.

[Huang and Yu, 1999] Yun-Wu Huang and Philip S. Yu. Adaptive query processing for time-series data. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining,* pages 282–286, 1999.

[Ikeda *et al*., 1999] Keita Ikeda, Bradley V. Vaughn, and Stephen R. Quint. Wavelet decomposition of heart period data. In *Proceedings of the IEEE First Joint BMES/EMBS Conference,* pages 3–11, 1999.

[Keogh and Pazzani, 1998] Eamonn J. Keogh and Michael J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *Proceedings of the Fourth ACM International Conference on Knowledge Discovery and Data Mining,* pages 239–243, 1998.

[Keogh and Pazzani, 2000] Eamonn J. Keogh and Michael J. Pazzani. Scaling up dynamic time warping for data mining applications. In *Proceedings of the Sixth ACM International Conference on Knowledge Discovery and Data Mining,* pages 285–289, 2000.

[Keogh *et al*., 2001] Eamonn J. Keogh, Selina Chu, David Hart, and Michael J. Pazzani. An online algorithm for segmenting time series. In *Proceedings of the IEEE International Conference on Data Mining,* pages 289–296, 2001.

[Keogh, 1997] Eamonn J. Keogh. Fast similarity search in the presence of longitudinal scaling in time series databases. In *Proceedings of the Ninth IEEE International Conference on Tools with Artificial Intelligence,* pages 578–584, 1997.

[Kim *et al*., 2001] Sang-Wook Kim, Sanghyun Park, and Wesley W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *Proceedings of the Seventeenth International Conference on Data Engineering,* pages 607–614, 2001.

[Lam and Wong, 1998] Sze Kin Lam and Man Hon Wong. A fast projection algorithm for sequence data searching. *Data and Knowledge Engineering,* 28(3):321–339, 1998.

[Last *et al*., 2001] Mark Last, Yaron Klein, and Abraham Kandel. Knowledge discovery in time series databases. *IEEE Transactions on Systems, Man, and Cybernetics,* Part B, 31(1):160–169, 2001.

[Lee *et al*., 2000] Seok-Lyonh Lee, Seok-Ju Chun, Deok-Hwan Kim, Ju-Hong Lee, and Chin-Wan Chung. Similarity search for multidimensional data sequences. In *Proceedings of the Sixteenth International Conference on Data Engineering,* pages 599–608, 2000.

[Li *et al*., 1998] Chung-Sheng Li, Philip S. Yu, and Vittorio Castelli. MALM: A framework for mining sequence database at multiple abstraction levels. In *Proceedings of the Seventh International Conference on Information and Knowledge Management,* pages 267–272, 1998.

[Lin and Risch, 1998] Ling Lin and Tore Risch. Querying continuous time sequences. In *Proceedings of the Twenty-Fourth International Conference on Very Large Data Bases,* pages 170–181, 1998.

[Lu *et al*., 1998] Hongjun Lu, Jiawei Han, and Ling Feng. Stock movement prediction and *N*-dimensional inter-transaction association rules. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery,* pages 12:1–7, 1998.

[Maurer and Dierks, 1991] Konrad Maurer and Thomas Dierks. *Atlas of Brain Mapping: Topographic Mapping of EEG and Evoked Potentials.* Springer-Verlag, Berlin, Germany, 1991.

[Niedermeyer and DaSilva, 1999] Ernst Niedermeyer and Fernando Lopes DaSilva. *Electroencephalography: Basic Principles, Clinical Applications, and*

Related Fields. Lippincott, Williams and Wilkins, Baltimore, MD, fourth edition, 1999.

[Park *et al*., 1999] Sanghyun Park, Dongwon Lee, and Wesley W. Chu. Fast retrieval of similar subsequences in long sequence databases. In *Proceedings of the Third IEEE Knowledge and Data Engineering Exchange Workshop,* 1999.

[Park *et al*., 2000] Sanghyun Park, Wesley W. Chu, Jeehee Yoon, and Chihcheng Hsu. Efficient searches for similar subsequences of different lengths in sequence databases. In *Proceedings of the Sixteenth International Conference on Data Engineering,* pages 23–32, 2000.

[Park *et al*., 2001] Sanghyun Park, Sang-Wook Kim, and Wesley W. Chu. Segment-based approach for subsequence searches in sequence databases. In *Proceedings of the Sixteenth ACM Symposium on Applied Computing,* pages 248–252, 2001.

[Perng *et al*., 2000] Chang-Shing Perng, Haixun Wang, Sylvia R. Zhang, and D. Scott Parker. Landmarks: A new model for similarity-based pattern querying in time series databases. In *Proceedings of the Sixteenth International Conference on Data Engineering,* pages 33–42, 2000.

[Policker and Geva, 2000] Shai Policker and Amir B. Geva. Non-stationary time series analysis by temporal clustering. *IEEE Transactions on Systems, Man, and Cybernetics,* Part B, 30(2):339–343, 2000.

[Pratt and Fink, 2002] Kevin B. Pratt and Eugene Fink. Search for patterns in compressed time series. *International Journal of Image and Graphics,* 2(1):89–106, 2002.

[Pratt, 2001] Kevin B. Pratt. Locating patterns in discrete time series. Master's thesis, Computer Science and Engineering, University of South Florida, 2001.

[Qu *et al*., 1998] Yunyao Qu, Changzhou Wang, and Xiaoyang Sean Wang. Supporting fast search in time series for movement patterns in multiple scales. In *Proceedings of the Seventh International Conference on Information and Knowledge Management,* pages 251–258, 1998.

[Sahoo *et al*., 1988] Prasanna K. Sahoo, S. Soltani, A. K. C. Wong, and Y. C. Chen. A survey of thresholding techniques. *Computer Vision, Graphics and Image Processing,* 41:233–260, 1988.

[Samet, 1990] Hanan Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading, MA, 1990.

[Sheikholeslami *et al*., 1998] Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of the Twenty-Fourth International Conference on Very Large Data Bases,* pages 428–439, 1998.

[Singh and McAtackney, 1998] Sameer Singh and Paul McAtackney. Dynamic time-series forecasting using local approximation. In *Proceedings of the Tenth IEEE International Conference on Tools with Artificial Intelligence,* pages 392–399, 1998.

[Stoffer, 1999] David S. Stoffer. Detecting common signals in multiple time series using the spectral envelope. *Journal of the American Statistical Association,* 94:1341–1356, 1999.

[Yaffee and McGee, 2000] Robert A. Yaffee and Monnie McGee. *Introduction to Time Series Analysis and Forecasting.* Academic Press, San Diego, CA, 2000.

[Yi *et al*., 2000] Byoung-Kee Yi, Nikolaos D. Sidiropoulos, Theodore Johnson, H. V. Jagadish, Christos Faloutsos, and Alexadros Biliris. Online data mining for co-evolving time series. In *Proceedings of the Sixteenth International Conference on Data Engineering,* pages 13–22, 2000.