Time-series Bitmaps: A Practical Visualization Tool for working with Large Time Series Databases

Nitin Kumar   Nishanth Lolla   Eamonn Keogh   Stefano Lonardi   Chotirat Ann Ratanamahatana

University of California - Riverside
Computer Science & Engineering Department
Riverside, CA 92521, USA
{nkumar, vlolla, eamonn, stelo, ratana}@cs.ucr.edu

Abstract

The increasing interest in time series data mining in the last decade has resulted in the introduction of a variety of similarity measures, representations and algorithms. Surprisingly, this massive research effort has had little impact on real world applications. Real world practitioners who work with time series on a daily basis rarely take advantage of the wealth of tools that the data mining community has made available. In this work we attempt to address this problem by introducing a simple parameter-light tool that allows users to efficiently navigate through large collections of time series. Our system has the unique advantage that it can be embedded directly into the any standard graphical user interface, such as Microsoft Windows, thus making deployment easier. Our approach extracts features from a time series of arbitrary length, and uses information about the relative frequency of its features to color a bitmap in a principled way. By visualizing the similarities and differences within a collection of bitmaps, a user can quickly discover clusters, anomalies, and other regularities within their data collection. We demonstrate the utility of our approach with a set of comprehensive experiments on real datasets from a variety of domains.

Keywords: Time Series, Chaos Game, Visualization.

1   Introduction

The increasing interest in time series data mining in the last decade has resulted the introduction of a variety of similarity measures/ representations/ definitions/ indexing techniques and algorithms (see, e.g., [1][2][4][6][17][21][22][24]). Surprisingly, this massive research effort has had little impact on real world applications. Examples of implemented systems are rare exceptions [25]. Cardiologists, engineers, technicians and others who work with time series on a daily basis rarely take advantage of the wealth of tools that the data mining community has made available. While it is difficult to firmly establish the reasons for the discrepancy between tool availability and practical adoption, the following reasons suggested themselves after an informal survey.

- Time series data mining tools often come with a bewildering number of parameters. It is not obvious to the practitioner how these should be set [23].
- Research tools often require (relatively) specialized hardware and/or software, rather than the ubiquitous desktop PC/windows environment that prevails.
- Many tools have a steep learning curve, requiring the user to spend many unproductive hours learning the system before any possibility of useful work.

In this work we attempt to address this problem by introducing a simple parameter-light tool that allows users to efficiently navigate through large collections of time series. Our approach extracts features from a time series of arbitrary length, and uses information about the relative frequency of these features to color a bitmap in a principled way. By visualizing the similarities and differences within a collection of these bitmaps, a user can quickly discover clusters, anomalies, and other regularities within their data collection.

While our system can be used as an interactive tool, it also has the unique advantage that it can be embedded directly into the any standard graphical user interface, such as Windows, Aqua, X-windows, etc. Since users navigate through files by looking at their icons, we decided to employ the bitmap representation as the icon corresponding to each time series. Simply by glancing at the icons contained in a folder of time series files, a user can quickly identify files that require further investigation. In Figure 1 we have illustrated a simple example[1].
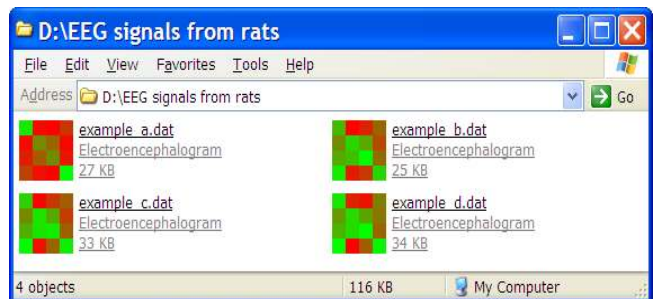

Figure 1: Four time series files represented as time series bitmaps. While they are all examples of EEGs, *example_a.dat* is from a normal trace, whereas the others contain examples of spike-wave discharges. The fact that there is some difference between one dataset and all the rest is immediately apparent from a casual inspection of the bitmap representation.

---

[1] This figure, and many others that follow in this work, suffer from monochromatic printing. We encourage the interested reader to visit [19] to view full color examples.

The utility of the idea shown in Figure 1 can be further enhanced by arranging the icons within the folder by *pattern similarity*, rather than the typical choices of arranging them by *size*, *name, date,* etc. This can be achieved by using a simple multidimensional scaling or a self-organizing map algorithm to arrange the icons.

Unlike most visualization tools which can only be evaluated subjectively, we will perform objective evaluations on the amount of useful information contained within a time series bitmap. More precisely, we will analyze the loss of accuracy of classification /clustering/anomaly detection algorithms when the input is based solely the information contained in the bitmap. As we will show, the experiments strongly confirm the utility of our approach.

The rest of the paper is organized as follows. In Section 2, we report on the related literature and review the SAX representation, which is a cornerstone of our approach. In Section 3 we introduce our general visualization technique called *Time Series Bitmaps*, and in Section 4 we consider *Time Series Thumbnails*, a special representation that can be tightly integrated into a standard graphical user interface. Section 5 sees a comprehensive empirical evaluation, and lastly, Section 6 offers conclusions and directions for future work.

2 Background and Related Work

We begin this section with a brief description of the classic time series data mining tasks. We review some of the relevant data visualization methods as part of the previous related work in this domain. Finally, we conclude this section by reviewing SAX, a symbolic representation which is a cornerstone to our scheme.

2.1 Time Series Data Mining Tasks

Until recently, the bulk of the time series data mining research was focused on the tasks of indexing [17][21][24], clustering [1][4][23] and classification [23]. In contrast, there has been relatively little work on time series visualization, in spite of the fact that the usefulness of visualization is well documented [27]. The few works on time series visualization tend to limit their attention to small datasets [25][32][14]. However, data visualization becomes of crucial importance when the size of the data is large. Because we claim that our visualization tool can be used in conjunction with some of the classic data mining tasks, we will briefly review them below.

2.1.1 Classification

The ability to predict the class of a previously unknown instance with the help of a training database is called classification. For example, imagine a scenario in which a patient visits a doctor because of chest pain and his/her ECG looks irregular. The doctor might want to search a database to find similar ECGs, in the hope that he/she will be able to offer clues about the patient's condition. The notion of similarity clearly depends on the particular measure chosen. For short time series, Euclidian distance [22] and Dynamic Time Warping (DTW) [1][20][30] are known to work exceptionally well. However for longer time series, the choice of the appropriate measure of similarity for classification tasks is still somewhat of an open question.

2.1.2 Clustering

Clustering is the problem of organizing data into classes such that there is high intra-class similarity and low inter-class similarity. It differs from classification in that the class labels are obtained directly from the data (cf. supervised vs. unsupervised learning). More informally, clustering finds a natural grouping among the objects according to the similarity measure chosen. Once again, for short time series such as gene expression data, Euclidian distance and DTW work very well, but for long time series, some model-based technique is typically used [11][18][34].

2.1.3 Anomaly Detection

The task of finding anomalies or irregularities in data has been an area of active research, which has long attracted the attention of researchers in biology, physics, astronomy, and statistics, in addition to the more recent work by the data mining community [23]. While the word "anomaly" implies that a radically different subsection of the data has been detected, we are interested in more subtle deviations in the data. The possibility of such subtle deviations from normality are reflected by the terms which are often used as synonyms for anomaly detection, such as interestingness/ deviation/ surprise/ deviation/ novelty detection, etc.

2.2 Related Work

There has been relatively little work done on visualizing massive time series data sets. We will briefly review some of the algorithms proposed in the literature and we will explain why they are not well suited to the task at hand.

The human eye is often heralded as the ultimate data-mining tool [27]. The ubiquitous time series plot is used to help visualize data in many aspects of human life such as medicine (ECG), finance (changes in stock market), business (profit-and-loss history of a company), aerospace (satellite data), meteorology (fluctuations in air temperature or pressure on a daily, monthly, or yearly basis)**,** entertainment (music, movies), etc. However many real world time series data sets are massive, and plots are limited by the resolution of the output device (screen, printer, etc.). Below we discuss some of the techniques used to mitigate the poor scalability of time series plots. These algorithms allow a greater scalability to larger datasets while leveraging off the human visual system to address the time series data mining tasks addressed above.

2.2.1 Arc Diagrams

Wattenberg introduced a visualization technique called Arc Diagrams [31] that exploits the fact that *"sequences often contain significant repeated subsequences"*. Many datasets come in the form of strings over finite alphabet (text, DNA, music, etc.). Datasets containing real-valued data points can be easily converted to a string using a discretization technique [26]. Once the input is available in the form of a string, translucent arcs are drawn between repeated substrings (as shown in the Figure 2).
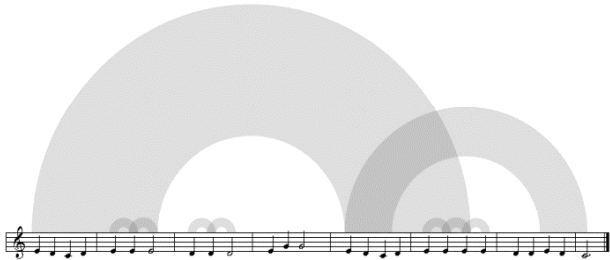

Figure 2: Arc diagram for a musical composition. The repetitive patterns are represented by translucent arcs.

The arc diagram allows the user to observe the occurrences of repeated patterns with a bird's eye view of the sequence's structure. However, this visualization technique is of less useful when there are many small repeated subsequences, which cause an uninformative jumble. This is often the case when the cardinality of the alphabet is small (e.g., DNA).

2.2.2 Spiral

Weber *et al.* presented an approach to visualize time series data based on a spiral representation [32]. The work in inspired by the observation that *"often, time series data such as temperature, radiation of light and economic cycles exhibit periodic structures"*. Each periodic structure of the time series is mapped on to a spiral ring to reveal the periodic behavior of the underlying process. The attributes are mapped to the properties of the spiral such as its color, texture and line thickness. A typical spiral diagram is shown in Figure 3.
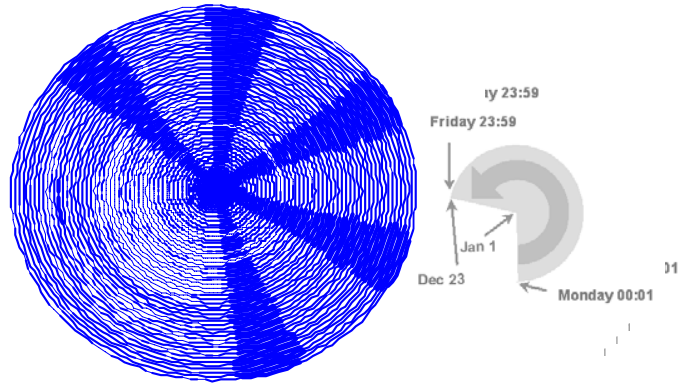

Figure 3: A spiral visualization of a power demand dataset.

Although Weber's technique is simple and intuitive, it is only useful for identifying patterns the time series that are periodic. Although spiral are more space effective than simple linear plots, scalability remains as an issue.

2.2.3 Viz-Tree

Lin *et al.* employed a visualization technique called Viz-Tree to discover patterns using a tree structure [25]. The time series data is first discretized to symbols using SAX (see below for a review of SAX). Then, the symbolic representation is encoded into a modified *suffix tree* wherein each branch of the tree represents one subsequence pattern. The frequencies of the various patterns are encoded as the line thickness of the suffix tree, so frequently occurring patterns show up as dense regions and rarely occurring patterns (possibly anomalies) show up as sparse lines. A screen shot of the Viz-Tree is shown in Figure 4.
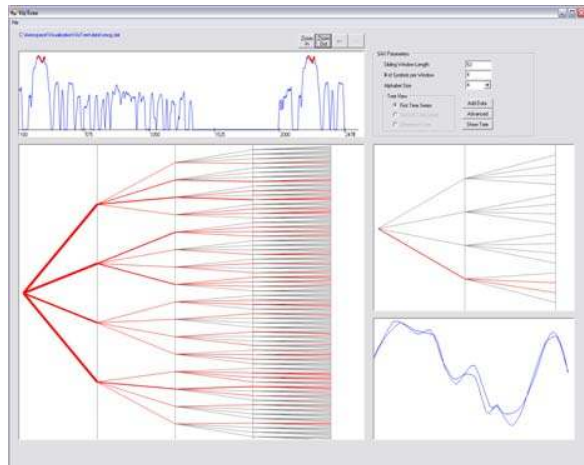

Figure 4: The Viz-Tree for an exhaust emissions dataset. Similar real valued time series (*bottom left*) tend to map to the same region of the tree (*middle left*) allowing a constant space summary of an arbitrarily long time series.

While the Viz-Tree is scalable to large data sets, it performs well only for certain data mining tasks like anomaly detection and motif discovery. In addition, it is demanding in terms of user training and computer resources.

2.3 Chaos Game Representations

Our visualization technique is partly inspired by an algorithm to draw fractals called the *Chaos game* [4]. The parameters of the game are defined by a set of pairs of linear equations (*i.e.,* an affine map), each of which is associated with a probability. Each affine map w_i is defined by six parameters

$$w_i \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix}$$

Formally, the set $\{w_1, w_2, ..., w_k\}$ and the associated probabilities defines an *iterated function system* (IFS). By changing the parameters of the IFS different fractals can be obtained (Sierpinski triangle, Dragon curve, Devil's staircase, etc).

As said, the chaos game is used to compute the fractal from the IFS. The game starts from a random point in the square *[0,1]×[0,1]*. At each step, a map is chosen at random according to its probability. The next point is obtained by applying the map to the previous point. Repeating this step thousands of times and plotting the trajectory of the points can visualize the fractal.

The chaos game representation has been used extensively to study DNA sequences (see, e.g., [3],[10],[12],[16]). Since DNA has a four letters alphabet, the most natural IFS is:

w	a	b	C	d	e	f	prob
1	.5	0	0	.5	0	0	.25
2	.5	0	0	.5	0	.5	.25
3	.5	0	0	.5	.5	0	.25
4	.5	0	0	.5	.5	.5	.25

If one runs the chaos game on this IFS using a truly random sequence, eventually all the points in *[0,1]×[0,1]* will be visited. In fact, the fractal associated with this IFS is the square *[0,1]×[0,1]*.

If instead of using the random number generator, one uses the sequence under study to drive the selection of the maps, the Chaos Game Representation (CGR) of the sequence is obtained [16],[17]. In Figure 5, we associated w_1 with the symbol "**A**", w_2 with **C**, w_3 with **G** and w_4 with **T**. When the choice of the map is driven by the sequence, it is easy to realize that each point in the square

[0,1]×[0,1] is in a one-to-one correspondence to a particular substring in the sequence.

Since we are limited by finite arithmetic, suppose that the square *[0,1]×[0,1]* is divided in $2^q \times 2^q$ pixels. Then, each pixel is uniquely identified by a substring of size q (see Figure 5). This representation is also called *quadtree* representation.

Based on the Suffix Theorem [16], it is easy to prove the following fundamental corollary (see also [3]).

Corollary. The number of occurrences of a substring in the original sequence is equal to the number of times the Chaos Games visits the pixel associated with that substring.

The method can produce a representation of DNA sequences, in which both local and global patterns are displayed. For example, a biologist can recognize that a particular substring, say in a bacterial genome, is rarely used. This would suggest the possibility that the bacteria have evolved to avoid a particular restriction enzyme site, which means that he might not be easily attacked by a specific bacterio-phage.

From our point of view, the crucial observation is that the CGR representation of a sequence allows the investigation of the patterns in sequences, giving the human eye a possibility to recognize hidden structures
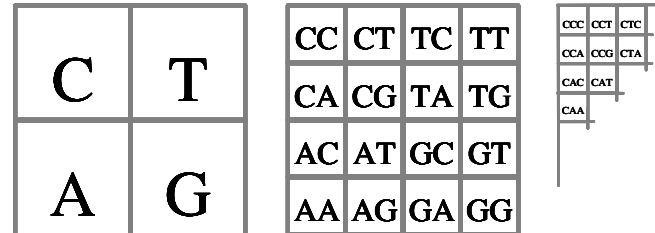

Figure 5: The quad-tree representation of a sequence over the alphabet {A,C,G,T} at different levels of resolution

We can get a hint of the potential utility of the approach if, for example, we take the first 16,000 symbols of the mitochondrial DNA sequences of four familiar species and use them to create their own file icons. Figure 6 below illustrates this. Even if we did not know these particular animals, we would have no problem recognizing that there are two pair of highly related species being considered.

With respect to the non-genetic sequences, Joel Jeffrey noted, *"The CGR algorithm produces a CGR for any sequence of letters"*[17]. However it is only defined for discrete sequences, and most time series are real valued.
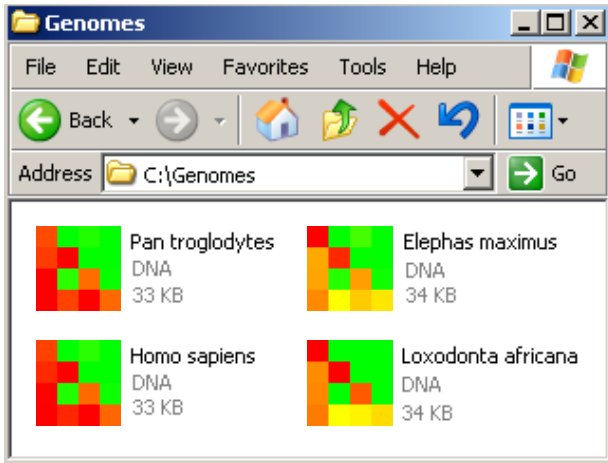
Figure 6: The gene sequences of mitochondrial DNA of four animals, used to create their own file icons using a chaos game representation. Note that *Pan troglodytes* is the familiar Chimpanzee, and *Loxodonta africana* and *Elephas maximus* are the African and Indian Elephants respectively. The file icons show that humans and chimpanzees have similar genomes, as do the African and Indian elephants.

This encouraged us to try a similar technique on time series data and investigate the utility of such representation on the classic data mining tasks of clustering, classification and visualization. Since CGR involves treating a data input as an abstract string of symbols, a discretization method is necessary to transform continuous time series data into discrete domain. For this purpose, we used the Symbolic Aggregate approXimation (SAX) [26], which we review below.

2.4 Symbolic Time Series Representations

While there are at least 200 techniques in the literature for converting real valued time series into discrete symbols [9], the SAX technique of Lin *et. al.* [26] is unique and ideally suited for data mining. SAX is the only symbolic representation that allows the lower bounding of the distances in the original space. The ability to efficiently lower bound distances is at the heart of hundreds of indexing algorithms and data mining techniques [2][8][20][22][26][30]. While we do not directly exploit the lower bounding property in this work, we note that if a representation is tightly lower bounding the original data, it must be representing it with great fidelity. It is this *implicit* property we are exploiting. We can be sure that the SAX representation is accurately summarizing the time series, and as we will show, a minor modification of the chaos game can accurately summarize the SAX sequences.

The SAX representation is created by taking a real valued signal and dividing it into equal sized sections. The mean value of each section is then calculated. By substituting each section with its mean, a reduced dimensionality piecewise constant approximation of the data is obtained.

This representation is then discretized in such a manner as to produce a word with approximately equi-probable symbols. Figure 7 shows a short time series being converted into the SAX word **baabccbc**.
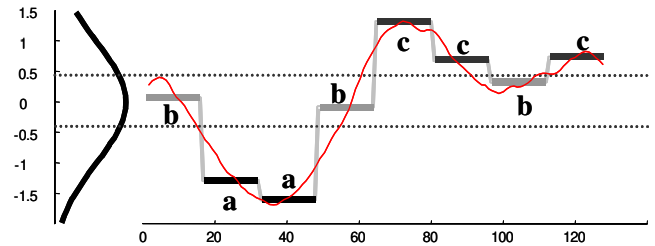

Figure 7: A real valued time series can be converted to the SAX word **baabccbc**. Note that all three possible symbols are approximately equally frequent.

The time and space complexity to convert a sequence to its SAX representation is linear in the length of the sequence.

It is very common to process very long time series. In that case, it is not necessarily a good idea to convert the entire time series into a *single* SAX word. For example, let us assume we have a time series that is composed of several sine waves. We would expect the SAX representation to be repetitive, something like **abcbabcbabcbabcba…** . However if we add a small linear trend to the entire sequence, the SAX representation would drastically change to something like **aaaaaaaabbbbbbbcc…** . In other words, the apparently *local* features of the sequence depend on the *global* structure. This is an undesirable property.

Therefore, for long time series, we slide a shorter window across it, and obtain a set of shorter SAX words. In the case of the sine waves example, we would expect to obtain a set of SAX words something like

> **abcba**
> **bcbab**
> **cbabc**
> **….**

Note that such a list is a *global* summary of *local* shapes. In the case above, consecutive rows only differ at the endpoints. For example, the **c** in the third place in the first word, moves to the second place in the second word, and the first place in the third word. In general, this is not always true. Because we are normalizing the contents of the sliding window at each step, a region that maps to one letter at the beginning of a word, may map to a different letter at the end of a word etc.

Although there is *some* redundancy between rows, only two bits are required per symbol (for an alphabet size four) and therefore this representation is much smaller than the original data.

Note that the user must choose both the length of the local sliding window *N*, and the number *n* of equal sized sections in which to divide it (as we will see, there is no choice to be made for alphabet size). A good choice for *N* should reflect the natural scale at which the events occur in the time series. For example, for ECGs this is about the length of one or two heartbeats. For traffic patterns, a 24-hour window makes sense. A good value for *n* depends of the complexity of the signal. Intuitively one would like to achieve a good compromise between fidelity of approximation and dimensionality reduction. Two groups of researchers have independently suggested using Minimum Description Length MDL to set these parameters [23][29]. As we shall see, the proposed technique is not too sensitive to parameter choices.

The SAX representation has been successfully used by various groups of researchers for indexing, classification, clustering [26], motif discovery [7][8][29], rule discovery, [28], visualization [25] and anomaly detection [23].

3 Time Series Bitmaps

At this point the connection between the two "ingredients" for the time series bitmaps should be evident. We have seen in Section 2.3 that the *Chaos game* [4] bitmaps can be used to visualize discrete sequences, and we have seen in Section 2.4 that the SAX representation is a discrete time series representation that has demonstrated great utility for data mining. It is natural to consider combining these ideas.

The *Chaos game* bitmaps are defined for sequences with an alphabet size of four. It is fortuitous that DNA strings have this cardinality. SAX can produce strings on any alphabet size. As it happens, a cardinality of four (or three) has been reported by many authors as an excellent choice for diverse datasets on assorted problems [7][8][23][25][26] [28][29].

We need to define an initial ordering for the four SAX symbols **a**, **b**, **c**, and **d**. We use simple alphabetical ordering as shown in Figure 8.

After converting the original raw time series into the SAX representation, we can count the frequencies of SAX "subwords" of length *L*, where *L* is the desired level of recursion. Level 1 frequencies are simply the raw counts of the four symbols. For level 2 we count pairs of subwords of size 2 (**aa**, **ab**, **ac**, etc). Note that we only count subwords taken from individual SAX words. For example, in the SAX representation in Figure 8 *middle right*, the last symbol of the first line is **a**, and the first symbol of the second word is **b**. However we do not count this as an occurrence of **ab**.
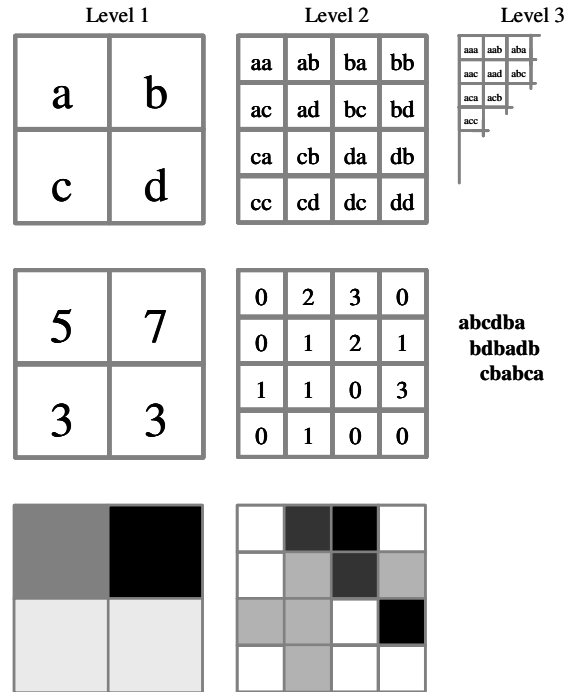

Figure 8: *Top*) The four possible SAX symbols are mapped to four quadrants of a square, and pairs, triplets etc are recursively mapped to finer grids. *Middle*) We can extract counts of symbols from a SAX representation and record them in the grids. *Bottom*) The recorded values can be linearly mapped to colors, thus creating a square bitmap.

Once the raw counts of all subwords of the desired length have been obtained and recorded in the corresponding pixel of the grid, a final step is required. Since the time series in a data collection may be of various lengths, we normalize the frequencies by dividing by the largest value. The pixel values thus range from 0 to 1. The final step is to map these values to colors. In the example above we mapped to grayscale, with 0 = *white*, 1 = *black*. However, it is generally recognized that grayscale is not *perceptually uniform* [33]. A color space is said to be perceptually uniform if small changes to a pixel value are approximately equally perceptible across the range of that value. For all images produced in this paper we use Matlab's "jet" color space, which is a linearization of a large fraction of all possible colors and which is designed to be perceptually uniform.

Note that unlike the arbitrarily long, and arbitrarily shaped time series from which they where derived, for a fixed *L*, the bitmaps have a constant space and structure.

We do not suggest any utility in viewing a single time series bitmap. The representation is abstract, and we do not expect a user to be able imagine the structure of time series given the bitmap. The utility of the bitmaps comes from the ability to efficiently compare and contrast them.

4 Time Series Thumbnails

A unique advantage of the time series bitmap representation is the fact that we can transparently integrate it into the user graphical interface of most standard operating systems.

Since most operating systems use the ubiquitous square icon to represent a file, we can arrange for the icons for time series files to appear as their bitmap representations. Simply by glancing at the contents of a folder of time series files, a user may spot files that require further investigation, or note natural clusters in the data.

The largest possible icon size varies by operating system. All modern versions of Microsoft Windows[2] support 32 by 32 pixels, which is large enough to support a bitmap of level 5. As we will see, level 2 or 3 seems adequate for most tasks/datasets.

To augment the utility of the time series bitmaps, we can arrange for their placement on screen to reflect their structure. Normally, file icons are arranged by one of a handful of common criteria, such as *name*, *date*, *size* etc.

We have created a simple modification of the standard Microsoft Windows (98 or later) file browser by introducing the concept of *Cluster View*. If *Cluster View* is chosen by the user, the time series thumbnails arrange themselves by similarly. This is achieved by performing Multi-Dimensional Scaling (MDS) of the bitmaps, and projecting them into a 2 dimensional space. For aesthetic reasons, we "snap" the icons to the closest grid point.

Figure 9 displays an example of *Cluster View* in Microsoft Windows XP Operating System.

In this example the *Cluster View* is obtained for five MIT-BIH Arrhythmia Database files. It is evident in the figure that *eeg1.dat*, *eeg2.dat* and *eeg3.dat* belong to one cluster whereas *eeg6.dat* and *eeg7.dat* belongs to another cluster. In this case, the grouping correctly reflects the fact that latter two files come from a different patient to first three.

To optimize the *Cluster View*, a cache (cluster.db) can be made to contain all relevant information required to generate the bitmaps and display their clustering, so that all the files do not have to be processed every time viewing the folder. However, even on a large screen full of small icons, an efficient MDS implementation can dynamically adjust the position of the icons in real time as the user changes the aspect ratio of the file browser.
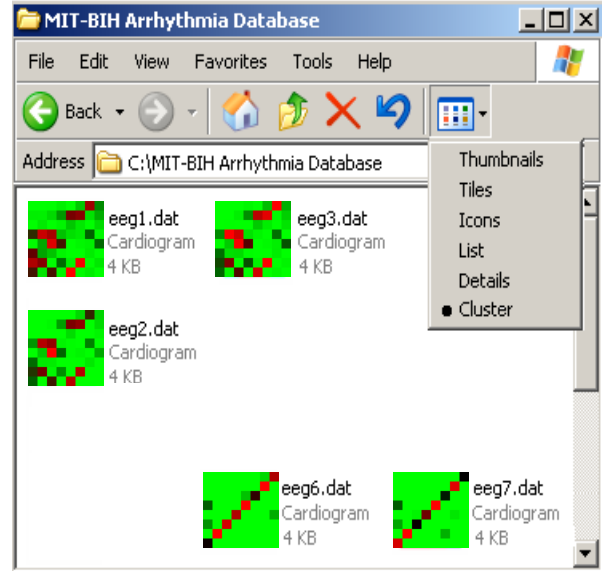
---

[2] Actually, Windows systems support icons of size 48 by 48 pixels, however we require a size that is an integer power of two.

Figure 9: A snapshot of a folder containing cardiograms when its files are arranged by "Cluster" option. Five cardiograms have been grouped into two different clusters based on their similarity.

5 Experimental Evaluation

In this section, we test our proposed approach with a comprehensive set of experiments. In Section 5.1 we will show a simple experiment which require subjective evaluation, but which strongly hint at the value of our approach. In Section 5.2 we will show some experiments that objectively measure the utility of our approach on classification, clustering and anomaly detection. We note once again that the quality of illustrations here suffers from monochromic printing and small-scale reproduction. We urge the interested reader to consult [19] for large-scale color reproductions and additional details.

5.1 Subjective Demonstration

First, we used the tool to browse the hundreds of datasets in the UCR archive. One such dataset, known as *Kalpakis_ECG*, contains 70 ECGS used to test an ARIMA based clustering technique [18]. When we glanced at this dataset with our tool we noticed something interesting. While ECGs (and therefore the thumbnails derived from them) can have great variability, five of the 70 thumbnails had radically different thumbnails. Figure 11 illustrates this on a subset of the full database.

It was natural to ask why this should be, so we further examined the original raw data, and noticed that the 5 relevant time series had radically different structure to all the rest.
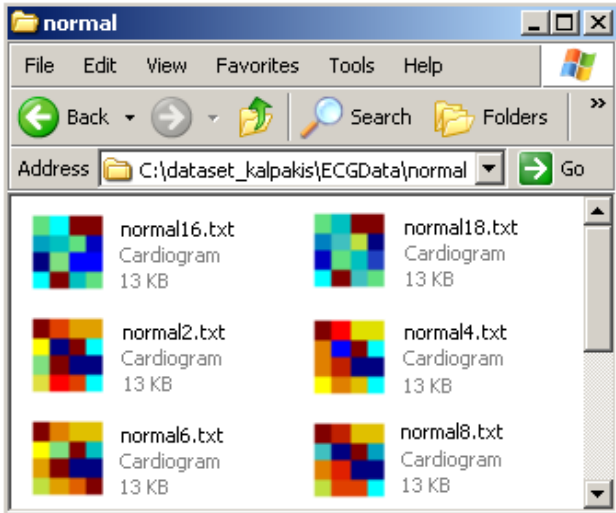
Figure 10: The two thumbnails in the first row are radically different from others. Whereas *normal2.txt, normal4.txt, normal6.txt* and *normal8.txt* are actually ECGs, *normal14.txt* and *normal16.txt* turned out to be the action potential of a pacemaker cell.

We asked a cardiologist to explain these findings. She informed us that the 5 recordings in question are not ECGs! They are in fact examples of the action potential of a normal pacemaker cell (not to be confused with the electronic man-made devices which mimic them, and are named after them).
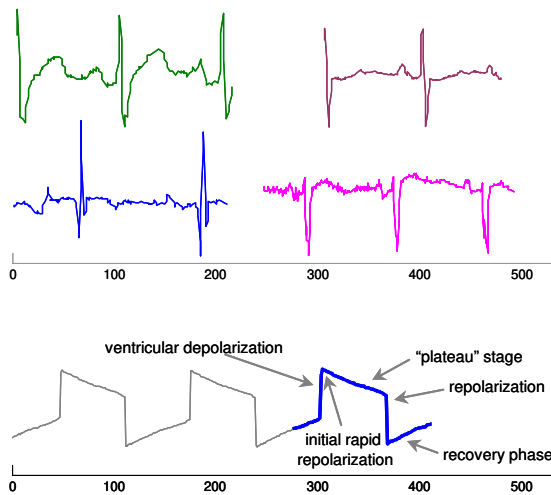

Figure 11: *Top*) Four snippets from randomly chosen ECGs from the Kalpakis_ECG dataset. Note that ECGs can have great variability. *Bottom*) A snippet from the *normal18* "ECG" from the Kalpakis_ECG dataset. In fact, this is not an ECG, but an example of the action potential of a normal pacemaker cell. The fact that this time series did not belong with the others was discovered by a casual glance with our time series thumbnail tool (See Figure 10).

Once the time series is plotted, it becomes obvious that these data are not ECGs. However, neither the original

authors [18] nor the other researchers who have published on this data [] seem to have noticed this. This suggests the utility of our approach a simple sanity check for practitioners working with large datasets.

5.2 Objective Experiments

As noted in Section 3, our time series bitmap representation has a unique feature among visualization techniques in that it allows the calculation of distance between two time series. This ability is particularly attractive because it allows very efficient comparison. Once the bitmap is created (in time linear in the length of the sequence) the time complexity for comparison is only O(L), since L is a small constant, this is O(1).

Note that we are mostly interested in relatively long time series here. For short time series, it has been forcefully shown that Euclidean distance and DTW are very hard to beat [22][30]. In any case, since short time series can be visualized directly, there is less motivation to use visualization techniques on them. It is difficult to define "short" and "long" formally. Intuitively, short time series would include things like a gene expression profiles (10 to 40 datapoints) and individual heartbeats (100 to 1,000 datapoints). In contrast long time series include things like a 5-minute trace of an ECG or 10-days worth of a telemetry sensor.

Since a distance/similarity measure is all that is required for classification, clustering and anomaly detection, we will compare time series thumbnails to classic solutions to these problems below.

5.2.1 Clustering

For our first experiment, we examined the UCR Time Series Archive for datasets that come in pairs. For example, in the **Buoy Sensor** dataset, there are two time series, *North Salinity* and *East Salinity*, and in the **Exchange Rate** dataset, there are two time series, *German Marc* and *Swiss Franc*. We were able to identify fifteen such pairs, from a diverse collection of time series covering the domains of finance, science, medicine, industry, etc. Although our method is able to deal with time series of different lengths, we truncated all time series to length 1,000 for visual clarity.

While the correct hierarchical clustering at the top of the tree is somewhat subjective, at the lower level of the tree, we would hope to find a single bifurcation separating each pair in the dataset. Our metric, Q, for the quality of clustering is therefore the number of such correct bifurcations divided by fifteen, the number of datasets. For a perfect clustering, $Q = 1$.

We compared to two well-known and highly referenced techniques, Markov models [11] and ARIMA models [18][34]. For each technique we spent one hour searching over parameter choice and reported only the best

performing result. To mitigate the problem of overfitting, we set the parameters on a different, but similar dataset.

The results for the three approaches are given in Table 1.

Table 1: The quality of clustering obtained by the 3 algorithms under consideration.

Algorithm	Q
Thumbnails	0.93
Markov Model	0.46
ARMA models	0.40

Figure 12 shows the resulting dendrogram for our approach. The dendrograms for the other approaches are omitted here for brevity, but may be viewed at [19].
We wished to test whether our approach was sensitive to its parameters, so we randomly changed them and reevaluated the quality of the clustering. Table 2 contains the results.

Table 2: The quality of clustering obtained by our approach with different parameter settings.

N	n	Q
64	8	0.86
77	11	0.73
54	9	0.86

These results suggest that our approach is not overly sensitive to parameter choices.

We repeated the experiment with a homogenous datasets. We considered a four-class ECG clustering problem, where each class corresponds to a different patient. Figure 13 shows the clustering obtained with level 3 bitmaps, using parameters $N = 50$, $n = 10$. The results are correct, in that each time series from a given patient is assigned to it own sub-tree. For this problem we found that we could vary the N amd n parameters by a factor of 4 ($N > n$) and still obtain the correct clustering.
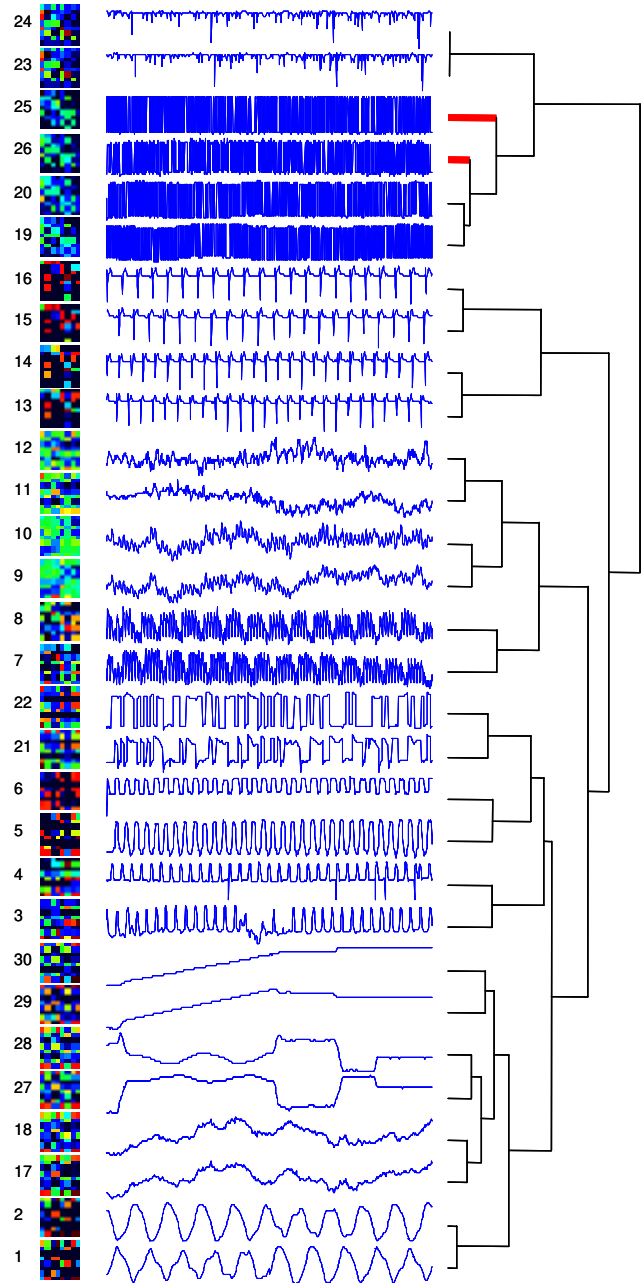

Figure 12: The clustering obtained by the time series thumbnail approach on a heterogeneous data collection. Bold lines denote incorrect subtrees. A key the data appears in Appendix A.
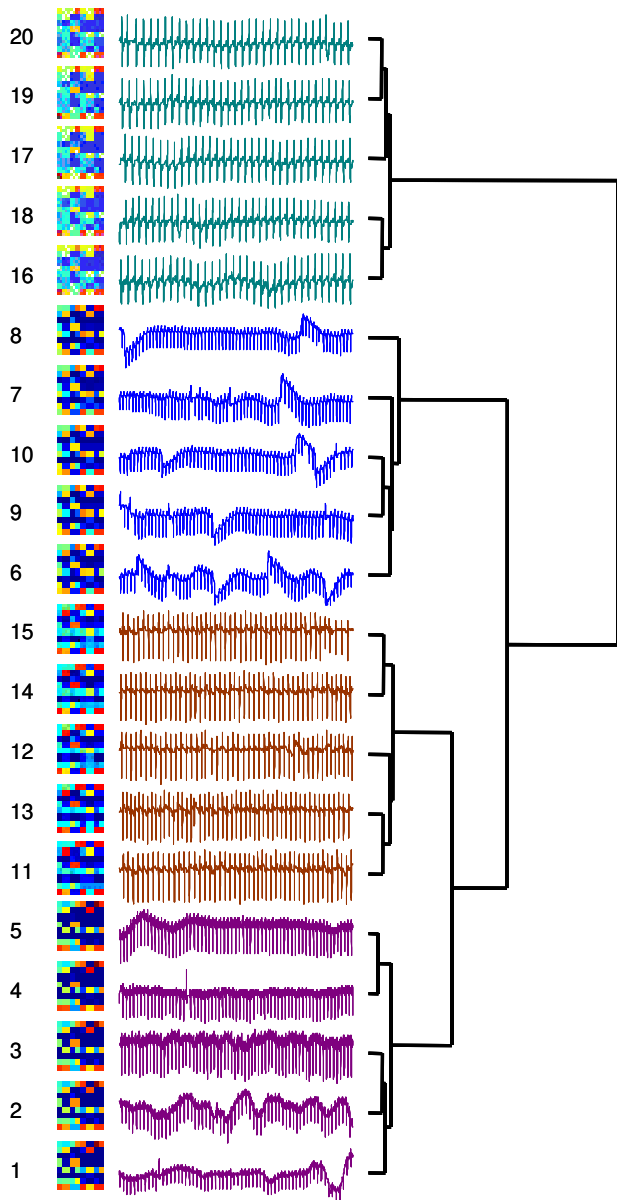
Figure 13: The clustering obtained by the time series thumbnail approach on a homogeneous data collection.

5.2.2 Classification

For classification we considered an ECG classification problem, and a video surveillance problem. Our ECG dataset is a four-class problem derived from BIDMC Congestive Heart Failure Database of four patients. Each instance consists of 3,200 contiguous data points (about 20 heartbeats) randomly extracted from a long (several hours) ECG signal. Twenty instances are extracted from each class (patient).

The video surveillance dataset is a time-series dataset extracted from video sequences of two actors either aiming a gun or simply pointing at a target. We randomly extracted twenty instances of 1,000 contiguous data points from each of the following long time series:

A. Male Actor 1 with gun
B. Male Actor 1 without gun (point)
C. Female Actor 2 with gun
D. Female Actor 2 without gun (point)

The problem is therefore a four-class problem of differentiating each of the acts independently – A vs. B vs. C vs. D. In total, each dataset contains eighty instances.

We compared to the ubiquitous Euclidean distance [21] [22][24] and DTW [20][30].

For both datasets we measure the error rates, using the one-nearest-neighbor with leaving-one-out evaluation method. The results are summarized in Table 3.

Table 3: Classification error rates for two datasets.

	Euclidean	DTW	Bitmaps
ECG	42.25 %	16.25 %	7.50 %
Surveillance	37.50 %	12.5 %	8.75 %

We also considered a *Normal* vs *Arrhythmia* problem that appeared in [11]. Using Markov models, the authors reported an error rate of 2%. With our technique, under virtually any parameter settings we achieve 0% error. We can achieve perfect classification using one nearest neighbor as above, or we can use MDS to project the data into 2 dimensional space and achieve perfect classification using a simple linear classifier, a decision tree or SVD. Figure 14 shows the data projected into 2D space, and the linear classifier learned.
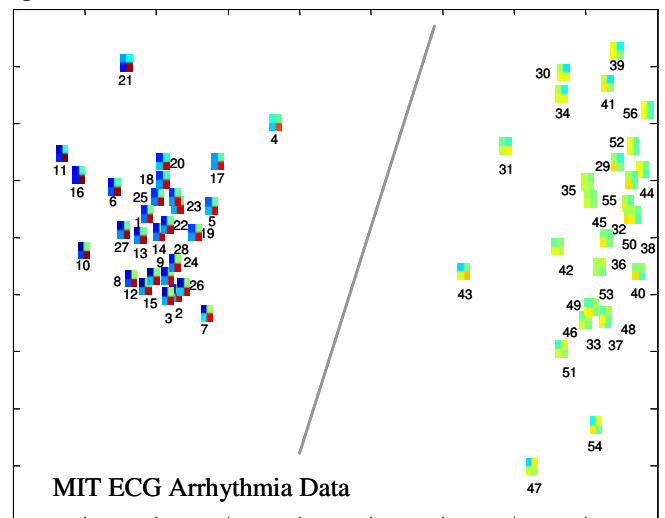

Figure 14: The MIT ECG Arrhythmia dataset projected into 2D space using only the information from a level 2-time series bitmap. The two classes are easily separated by a simple linear classifier (gray line).

5.2.3 Anomaly detection

The time series bitmap distance measure allows the creation of a simple anomaly detection algorithm.

We can create two concatenated windows, and slide them together across the sequence. At each time instance we build a time series bitmap for the two windows, and measure the distance between them. This distance we report as an anomaly score. Figure 15 illustrates the idea on some annotated ECG data.
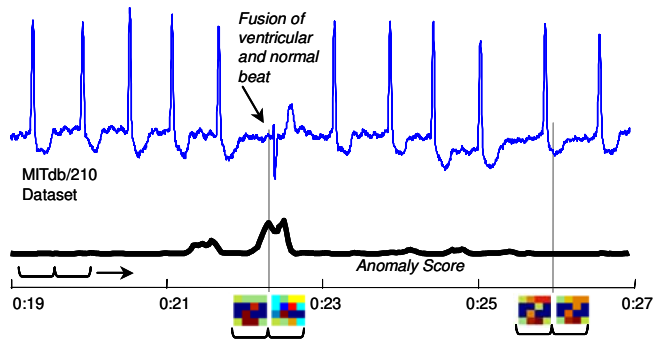

Figure 15: Using time series bitmaps as anomaly detectors. *Top*) A subsection of an ECG dataset. A cardiologist annotated an anomaly at approximately the 22-second mark. *Bottom*) The score for our approach shows a strong peak for the duration of the anomaly.

This approach easily detects the single anomaly shown, and the rest of the annotated anomalies in this dataset (not shown). At each "step" of the sliding window we can incrementally ingress a new data point, and egress an old data point (updating only two pixels of the each thumbnail), so the time complexity is linear in the length of the time series.

In the simple example above, both sliding windows are of the same length. More generally, one may wish for the trailing window to be larger, so that it retains more of a "memory" of the previous data. We leave such considerations for future work.

6   Conclusions and Future Work.

In this work, we have introduced a new framework for visualization of time series. Our approach is unique in that it can be directly embedded into any standard GUI operating system. We demonstrated the effectiveness of our approach on a variety of tasks and domains. Future work includes an extensive user study, and investigating techniques to automatically set the system parameters.

References

[1] Aach, J., & Church, G. (2001). *Aligning gene expression time series with time warping algorithms*. Bioinformatics, Volume 17, pp. 495-508.

[2] Agrawal, R., Lin, K. I., Sawhney, H. S., & Shim, K. (1995). *Fast similarity search in the presence of noise, scaling, and translation in times-series databases*. In Proceedings of twenty-first International Conference on Very Large Databases, pp. 490-501.

[3] Almeida, J.S., Carrico, J.A., Maretzek, A., Noble, P.A., & Fletcher, M. (2001). *Analysis of genomic sequences by Chaos Game Representation.* Bioinformatics, 17(5), pp. 429-37.

[4] Barnsley, M.F., & Rising, H. (1993). *Fractals Everywhere*, second edition, Academic Press.

[5] Bar-Joseph, Z., Gerber, G., Gifford, D., Jaakkola, T., & Simon, I. (2002). *A new approach to analyzing gene expression time series data*. In Proceedings of the sixth Annual International Conference on Research in Computational Molecular Biology, pp. 39-48.

[6] Berndt, D., & Clifford, J. (1994). *Using dynamic time warping to find patterns in time series*, AAAI Workshop on Knowledge Discovery in Databases, pp. 229-248.

[7] Celly, B. & Zordan, V. B. (2004). Animated People Textures. In proceedings of the 17th International Conference on Computer Animation and Social Agents. Geneva, Switzerland.

[8] Chiu, B., Keogh, E., & Lonardi, S. (2003). *Probabilistic Discovery of Time Series Motifs*. In the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 493-498.

[9] Daw, C. S., Finney, C. E. A. & Tracy, E. R. (2001). Symbolic Analysis of Experimental Data. Review of Scientific Instruments. (2002-07-22).

[10] Fiser. A., Tusnady, G.E., & Simon, I. (1994). *Chaos game representation of protein structures.* J. Mol. Graph, 12(4), pp. 302-4, 295.

[11] Ge, X., & Smyth, P. (2000). *Deformable Markov model templates for time-series pattern matching*. In proceedings of the sixth ACM SIGKDD, pp. 81-90.

[12] Hahn, M.W., Stajich, J.E., & Wray, G.A. (2003). *The Effects of Selection Against Spurious Transcription Factor Binding Sites*. Mol. Biol. Evol. 20(6), pp. 901-906.

[13] Haigh, K., Foslien, W., & Guralnik, V. (2004). *Visual Query Language: Finding patterns in and relationships among time series data*, In Proceedings

of the seventh Workshop on Mining Scientific and Engineering Datasets.

[14] Havre, S., Hetzler, E., Whitney, P., & Nowell, L. (2002). *ThemeRiver: Visualizing Thematic Changes in Large Document Collections.* IEEE Transactions on Visualization and Computer Graphics, pp. 9-20.

[15] Hu, N., Dannenberg, R.B., & Tzanetakis, G. (2003). *Polyphonic Audio Matching and Alignment for Music Retrieval,* IEEE Workshop on Applications of Signal Processing to Audio and Acoustics.

[16] Jeffrey, H.J. (1990). *Chaos game representation of gene structure.* Nucleic Acids Research 18, pp. 2163-2170.

[17] Jeffrey, H.J. (1992). *Chaos Game Visualization of Sequences.* Comput. & Graphics 16, pp. 25-33.

[18] Kalpakis, K., Gada, D., & Puttagunta, V. (2001). *Distance Measures for Effective Clustering of ARIMA Time-Series.* In the Proceedings of the 2001 IEEE International Conference on Data Mining, pp. 273-280.

[19] Keogh, E. www.cs.ucr.edu/~nkumar/SDM05.html

[20] Keogh, E. (2002). *Exact indexing of dynamic time warping.* In Proceedings of the twenty-eighth International Conference on Very Large Data Bases, pp. 406-417.

[21] Keogh, E., Chakrabarti, K., Pazzani, M., & Mehrotra (2001). *Locally adaptive dimensionality reduction for indexing large time series databases.* In Proceedings of ACM SIGMOD Conference on Management of Data, pp. 151-162.

[22] Keogh, E. & Kasetty, S. (2002). *On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration.* In the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 102-111.

[23] Keogh, E., Lonardi, S., & Ratanamahatana, C. (2004). *Towards Parameter-Free Data Mining.* In proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[24] Korn, F., Jagadish, H., & Faloutsos, C. (1997). *Efficiently supporting ad hoc queries in large datasets of time sequences.* In Proceedings of SIGMOD, pp. 289-300.

[25] Lin, J., Keogh, E., Lonardi, S., Lankford, J.P. & Nystrom, D.M. (2004). *Visually Mining and Monitoring Massive Time Series.* In proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining..

[26] Lin, J., Keogh, E., Lonardi, S. & Chiu, B. (2003) *A Symbolic Representation of Time Series, with Implications for Streaming Algorithms.* In proceedings of the eighth ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery.

[27] Shneiderman, B. (2002). *Inventing discovery tools: combining information visualization with data mining.* Information Visualization 1(1): 5-12.

[28] Silvent, A. S., Carbay, C., Carry, P. Y. & Dojat, M. (2003). *Data, Information and Knowledge for Medical Scenario Construction.* In proceedings of the Intelligent Data Analysis In Medicine and Pharmacology Workshop. Protaras, Cyprus.

[29] Tanaka, Y. & Uehara, K. (2004). *Motif Discovery Algorithm from Motion Data.* In proceedings of the 18th Annual Conference of the Japanese Society for Artificial Intelligence (JSAI). Kanazawa, Japan.

[30] Ratanamahatana, C.A., & Keogh, E. (2004). *Everything you know about Dynamic Time Warping is Wrong.* 3rd Workshop on Mining Temporal and Sequential Data, in conjunction with the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[31] Wattenberg, M. (2002). *Arc Diagrams: Visualizing Structure in Strings.* In proceedings of the IEEE Symposium on Information Visualization, pp. 110-116.

[32] Weber, M., Alexa, M., & Mueller, W. (2001) *Visualizing Time-Series on Spirals.* In proceedings of the IEEE Symposium on Information Visualization,

[33] Wyszecki, G. (1982). *Color science: Concepts and methods, quantitative data and formulae,* 2nd edition. New York, Wiley, 1982.

[34] Xiong, Y., & Yeung, D.Y. (2002). *Mixtures of ARMA models for model-based time series clustering.* In Proceedings of the IEEE International Conference on Data Mining, pp.717-720.

Appendix A: Key to Datasets

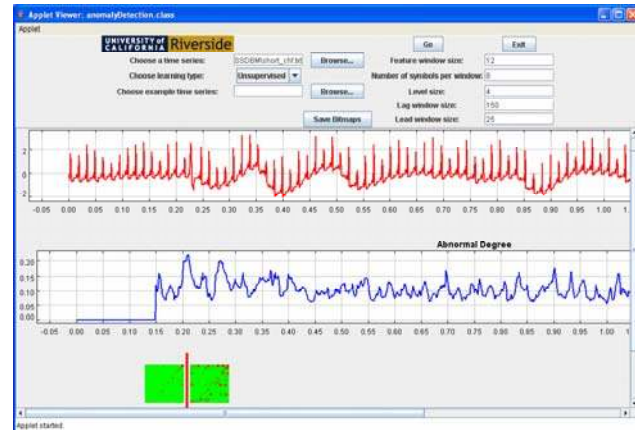Table A: The datasets used in the Experiments in Section 5.1.1

1	MotorCurrent1:	17	Exchange Rate: Swiss Franc
2	MotorCurrent2:	18	Exchange Rate: German Mark
3	Video Surveillance: Ann, gun	19	Furnace: heating input
4	Video Surveillance: Ann, no gun	20	Furnace: cooling input
5	Video Surveillance: Eamonn, gun	21	Reel 2: angular speed
6	Video Surveillance: Eamonn, no gun	22	Reel 2: tension
7	Power Demand: Jan-March (Italian)	23	Balloon1
8	Power Demand: April-June (Italian)	24	Balloon2 (lagged)
9	Great Lakes (Erie)	25	Evaporator: feed flow
10	Great Lakes (Ontario)	26	Evaporator: vapor flow
11	Buoy Sensor: North Salinity	27	Shuttle Inertia Sensor X
12	Buoy Sensor East Salinity	28	Shuttle Inertia Sensor X
13	Koski ECG: slow 1	29	Shuttle Inertia Sensor Z
14	Koski ECG: slow 2	30	Shuttle Inertia Sensor Z
15	Koski ECG: fast 1		
16	Koski ECG: fast 2		

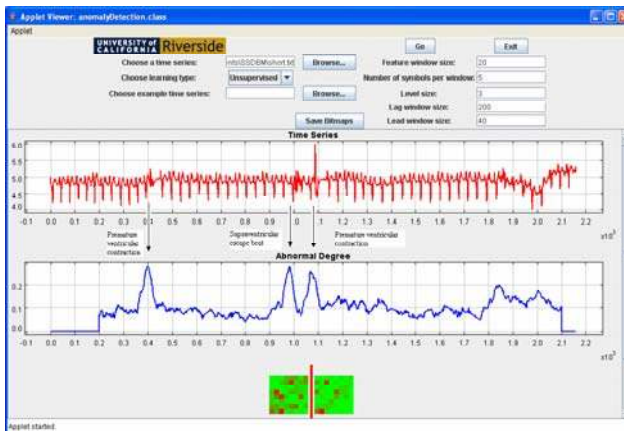## Appendix B: Additional Anomaly Detection Results

Note that these results do *not* appear in the SIAM paper.

A: Using time series bitmaps as an anomaly detector. *Top*) A subsection of an ECG dataset. A cardiologist annotated a premature ventricular contraction at approximately the 1.4 mark. *Bottom*) The score for our approach shows a strong peak for the duration of the anomalous heartbeat.

B: A cardiologist annotated two premature ventricular contractions at approximately the 0.4 and 1.1 mark respectively, and a supraventricular escape beat at about the 1.0 mark. Our approach easily detects all the three anomalies. *Top*) A subsection of an ECG dataset. *Bottom*) The score for our approach shows three strong peaks for the duration of the anomalous heartbeat.

C: This figures shows a very complex and noisy ECG. But according to a cardiologist, there is only one abnormal heart beat at approximately the 0.23 mark. Our tool easily finds it.