A Multiresolution Symbolic Representation of Time Series

Vasileios Megalooikonomou[1] Qiang Wang[1] Guo Li[1] Christos Faloutsos[2]
[1]*Department of Computer & Information Sciences*
Temple University
Philadelphia, PA, USA
{vasilis,qwang,gli}@temple.edu
[2]*Department of Computer Science*
Carnegie Mellon University
Pittsburgh, PA, USA
christos@cs.cmu.edu

Abstract

Efficiently and accurately searching for similarities among time series and discovering interesting patterns is an important and non-trivial problem. In this paper, we introduce a new representation of time series, the Multiresolution Vector Quantized (MVQ) approximation, along with a new distance function. The novelty of MVQ is that it keeps both local and global information about the original time series in a hierarchical mechanism, processing the original time series at multiple resolutions. Moreover, the proposed representation is symbolic employing key subsequences and potentially allows the application of text-based retrieval techniques into the similarity analysis of time series. The proposed method is fast and scales linearly with the size of database and the dimensionality. Contrary to the vast majority in the literature that uses the Euclidean distance, MVQ uses a multi-resolution/hierarchical distance function. We performed experiments with real and synthetic data. The proposed distance function consistently outperforms all the major competitors (Euclidean, Dynamic Time Warping, Piecewise Aggregate Approximation) achieving up to 20% better precision/recall and clustering accuracy on the tested datasets.

1. Introduction

The problem of efficient retrieval of similar time series has received a lot of attention due to its many applications in different domains. Briefly, this problem can be stated as follows: Given a query sequence q, a database S of N sequences, $S_1,S_2,\ldots,S_N$, a distance measure D and a tolerance threshold ε, find the set of sequences R in S that are within distance ε from q. More precisely, find: $R = \{S_i \in S \mid D(q, S_i) \leq \varepsilon\}$.

To compare two given time series, a suitable measure of similarity should be given. Naive approaches for comparing time sequences generally take polynomial time in the length of the sequences, typically linear or quadratic time. These approaches are not useful for large time series databases. Promising techniques include those that are based on the reduction of dimensionality of the original sequences. In this case, the sequences can be represented as multidimensional vectors and similar sequences can be retrieved in sublinear time.

There may be several different criteria to evaluate a method, but generally speaking, a good one should be fast, scalable, and accurate (according to some ground truth). In this paper, we introduce a new method that satisfies these requirements. Our method is called Multiresolution Vector Quantized (MVQ) approximation and has the following characteristics:

1) It uses time-tested 'vector quantization' methods to discover a 'vocabulary' of subsequences;
2) It takes multiple resolutions into account – this brings improved accuracy;
3) It provides a new distance function utilizing text-based techniques from Information Retrieval, to weigh down uninteresting matches, thus improving the accuracy.

As Agrawal et al. [2] proposed, compared to the Euclidean distance, a more intuitive idea is that two series should be considered similar if they have enough non-overlapping time-ordered pairs of subsequences that are similar. In this paper, instead of calculating the Euclidean distance, we first extract key subsequences utilizing the Vector Quantization (VQ) [12] technique and encode each time series based on the frequency of appearance of each key subsequence. We then calculate similarities between different time series in terms of key subsequence matches. This method can be very meaningful in many domains, for example, when comparing two stocks during a long period, we may want to find out during how many months the stocks have similar movements, though the same trend may appear in different months for different stocks. This application is similar to mining motifs in massive time series databases [22].

While the histogram metric can record the local information very well, it may lose much global information of the time series, since it does not keep track of the order of appearance of different key subsequences. To deal with this problem, we propose to apply a hierarchical mechanism: the original time series are processed at several different resolutions, and similarity analysis is performed using a weighted distance function combining all the resolution levels. For example, when considering a time series representing a stock price

movement, we know that subsequences of different length have different real meanings. If the length is 5, the subsequence stands for a weekly trend of the stock. Similarly for length 20 we have the monthly trend.

As we demonstrate in the experiments, MVQ outperforms previous state of the art methods in clustering and similarity searches. Intuitively, the excellent performance of the proposed method can be justified because of the following facts:

1) it exploits prior knowledge about the data using a learning approach
2) it takes multiple resolutions into account and
3) unlike wavelets (that also take multiple resolutions into account) it partially ignores the ordering of the 'codewords' within the time sequence due to the histogram model that is being used to calculate similarity.

Moreover, the proposed representation is symbolic employing key subsequences and allows the application of text-based retrieval techniques into the similarity analysis of time series.

2. Background

2.1 Related Work

Many approaches and techniques have been proposed in the past decade [1, 2, 4, 9, 10, 13, 14, 16, 18, 19, 21, 27, 31, 32] that address the problem of similarity in time series.

To deal with dimensionality reduction, the solution to extract a signature from each sequence and to index the signature space was originally proposed by Faloutsos et al. [9,10]. To guarantee completeness (i.e., no false dismissals) the admissibility criterion that the distance function used in the signature space must underestimate the true distance measure (bounding lemma) was also proposed [10]. Obeying the admissibility criterion, many methods have been suggested and proved useful in different fields, such as the F-index introduced by Agrawal et al. [1] or the ST-index proposed by Faloutsos et al. [10].

Other approaches for efficient similarity searches on time sequences are based on piecewise constant approximation (PCA) or piecewise aggregate approximation (PAA). Yi and Faloutsos [32] and Keogh et al. [19,21] proposed to divide each sequence into k segments of equal length and to use the average value of each segment as a coordinate of a k-dimensional feature vector. The advantages of this transform are that it is very fast and easy to implement, the signature can be used with arbitrary L_p norms, and the index can be build in linear time. In addition, the representation can be used with a weighted Euclidean distance where each segment of the sequence has different weight. Keogh et al. [18] have also proposed an Adaptive Piecewise Constant Approximation (APCA) where the segments can be of variable length offering a more effective compression than PCA. In [26] the authors propose a piecewise vector quantized approximation (PVQA) of time series. In [7] a technique for compressing multiple streams of data in sensor networks that employs an approximate representation using a base signal extracted from historic information has been proposed. The algorithm constructs a "dictionary" of candidate base signals in the process of building a base signal. The use of multi-scale histograms and a weighted Euclidean distance for measuring the similarity of time series at several precision levels has been investigated in [6]. In addition, general dimensionality reduction techniques such as Singular Value Decomposition (SVD) have been used in time series data [19].

For these methods in which the distance metric lower bounds the Euclidean distance, one of the most significant characteristics is the avoidance of false dismissals, though there may be a lot of false alarms. However, in some cases, the existence of too many false alarms may decrease the efficiency of retrieval. At the same time, as many researchers have mentioned in their work [15,29], the Euclidean distance is not always the optimal distance measure. For example, in some time series, different parts have different levels of significance in their meaning. Also, the Euclidean distance does not allow shifting in time axis, which is not unusual in real life applications. In order to extract high-level features out of time series, Koudas et al. [28] formalized problems of identifying various "representative" trends in time series data. Since the Euclidean is not the best distance one can use (as shown later in our paper and in papers we referenced earlier), here, we propose a new distance function. We do not deal with the problem of lower bounding the Euclidean on the original vectors since this is not so meaningful anymore.

2.2 Preliminaries

To make the presentation of the proposed work clear, we now give descriptions of various concepts and definitions used in the paper. We start with the definition of a time sequence and its subsequences.

Definition 1. Time Sequence: A sequence (ordered collection) of real values. $X = x_1, x_2,\ldots, x_n$, where n can be very large.

Definition 2. Subsequence: Given a time sequence $X = x_1, x_2,\ldots, x_n$, of length n, a subsequence S of X is a sequence of length m consisting of contiguous positions from X, i.e., $S=x_k,x_{k+1},\ldots,x_{k+m-1}$; $1 \le k \le n-m+1$.

In similarity analysis, we need to define a metric for the similarity, that is, a measure of the distance between two time series. Given two time series, $X = x_1, x_2,\ldots, x_n$, $Y = y_1, y_2, \ldots, y_n$, their distance, D, is defined, in general,

as an L_p norm, where for p=2, the distance is the Euclidean, the most popular among the metrics. An intuitive notion of exact and approximate similarity was also formalized by Goldin, and Kanellakis [8].

Obviously, the simplest way of calculating the similarity (or distance) among time series is to compute the Euclidean distance directly, i.e., on the original series. For a small dataset this may be feasible, however, for large data sets efficiency is a problem, since the time complexity is O(N*n), where n is the number of features that need to be represented for each time series and N is the number of time series in the dataset. In order to compute efficiently while keeping the accuracy not significantly affected, many techniques of dimensionality reduction (as introduced in section 2.1) have been suggested.

In addition to the computational complexity associated with the Euclidean distance calculation on the original time series, we cannot always be sure that the nearest neighbors in Euclidean space are indeed the most similar ones. This is because the point-based information model (computing similarity based on every point) contains only low-level features of the time series and it is vulnerable to different kinds of shape transformations, such as shifting and scaling. Under such circumstances, it would be better if we could find some high-level features and apply a more robust information retrieval model for time series analysis.

Based on this idea, we introduce a new framework that uses key subsequences to represent time series and facilitate similarity retrieval. This framework consists of the following main components:

1) Codebook generation from a set of training samples;
2) Time series encoding using the codebook;
3) Time series feature representation and retrieval.

This framework is similar to the key block framework suggested by Zhang et al. [33] for content-based image retrieval. In the time sequences domain the idea was introduced in [26]. However, in order to keep both local and global information and improve the accuracy, we introduce the use of multiple codebooks with different resolutions. For each resolution, Vector Quantization [12, 24] is applied to discover the vocabulary of subsequences in a time series database.

In VQ a codeword (or codevector) is used to represent a number of similar vectors. More precisely, a vector quantizer Q of dimension n and size s is a mapping: Q: $\Re^n \rightarrow C$ from a vector or a point in n-dimensional Euclidean space, $\Re^n$, to a finite set $C=\{c_1, c_2, \ldots, c_s\}$, the codebook, containing s output or reproduction points $c_i \in \Re^n$, called codewords. Associated with every s-point VQ is a partition of $\Re^n$ into s regions or cells R_i for $i \in J \equiv \{1,2,\ldots,s\}$ where $R_i=\{x \in \Re^n: Q(x)=c_i\}$. For a given

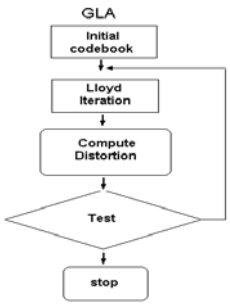


Figure 1. The Generalized Lloyd Algorithm (GLA).

distortion function[1] $d(x,c_i)$ (such as the mean squared error (MSE)) between an input vector x and a codeword c_i, an optimal mapping should satisfy two conditions:

(a) Nearest neighbor Condition (NNC): For a given codebook, the optimal partition $R = \{R_i: i=1,2,\ldots,s\}$ satisfies: $R_i = \{x : d(x,c_i) \le d(x,c_j); \forall j\}$

where c_i is the codeword representing partition R_i. Given a point x in the dataset, the encoding function for x, $Encoding(x)=c_i$ only if $d(x,c_i) \le d(x,c_j)$ $\forall j$.

(b) Centroid condition (CC): For a given partition region $\{R_i: i=1,\ldots,s\}$ the optimal reconstruction vector (codeword) satisfies: $c_i=centroid(R_i)$ where the centroid of a set $R=\{x_i: i=1,\ldots|R|\}$ is defined as:

$$centroid(R) = \frac{1}{|R|}\sum_{i=1}^{|R|} x_i$$

The Generalized Lloyd Algorithm (GLA) [24, 25] is an iterative procedure that produces a "locally optimal" codebook from a training set based on these two conditions (that form the Lloyd iteration). This is done during a training phase. The main structure of GLA is given in the flowchart (see Figure 1). Starting with an initial codebook, the GLA algorithm repeats the Lloyd iteration until the fractional drop of the distortion becomes less than a given threshold. This process is guaranteed to converge since from the necessary conditions for optimality each application of the Lloyd iteration must reduce or leave unchanged the average distortion [12].

To quantitatively measure the similarity between different time series encoded with a VQ codebook, we employ the Histogram Model (HM) that has been successfully applied in image retrieval [33]. We present this model in the context of time series analysis:

$$S_{HM}(q,t) = \frac{1}{1+dis(q,t)} \qquad (1)$$

where $dis(q,t) = \sum_{i=1}^{s} \frac{|f_{i,t} - f_{i,q}|}{1+f_{i,t}+f_{i,q}}$.

[1] The distortion is a measure of overall quality degradation due to approximation of a vector by its closest representative from a codebook.

In the formula, $f_{i,t}$ and $f_{i,q}$ refer to the appearance frequency of codeword c_i in time series t and q, respectively. Although this model focuses on the appearance of individual key subsequences in time series, correlation between key subsequences can also be addressed [33]. Information about some alternative models can be found in Appendix A.

3. Proposed Method: MVQ

We propose a new method to represent time series data, the Multiresolution Vector Quantized (MVQ) approximation, along with a new distance function. The method partitions each time series into equi-length segments and represents each segment with the most similar key subsequence from a codebook. The codebook is generated earlier during a training phase using VQ. By counting the appearance frequency of each codeword in each time series a new representation is obtained. The piecewise approximation with VQ encoding is applied at several resolutions. Table 1 gives a brief description of the notation we use in the rest of the paper. In the following subsections, we introduce the components of our method.

Table 1. Symbol Table

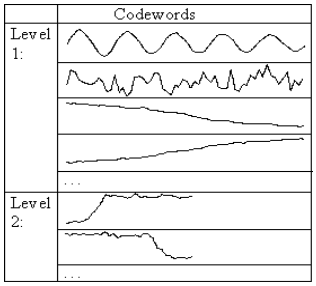
X	Original time series, $X= x_1,x_2,\ldots,x_n$ of length n
X'	Encoded form of the original time series $X'= f_1,f_2,\ldots,f_s$
N	Number of time series in the dataset
n	Length of original time series
C	Codebook: a set of codewords $\{c_1,\ldots,c_k,\ldots, c_s\}$
c	Number of resolution levels
s	Size of codebook
l	Length of codeword

3.1 Codebook Generation

For a given dataset, a codebook with s codewords $C=\{c_1,c_2, \ldots, c_s\}$ is first generated using a clustering algorithm (such as the GLA introduced in Section 2). We apply this algorithm to generate the codebook based on the dataset T of time series. The dataset is preprocessed before the generation of the codebook; each time series in T is partitioned into a number of segments each of length l and each segment forms a sample of the training set that is used to generate the codebook. Each codeword in the codebook corresponds to a key subsequence; it is an approximation for a certain group of subsequences of length l. All the time series in the database are then encoded using the codebook (see Section 3.2).

The version of GLA we use, requires a partition split mechanism to solve the initial codebook generation problem. The algorithm starts with a codebook containing only one codeword, the centroid of the whole training set. In each repetition and before the application of the Lloyd iteration, it doubles the number of codewords (and cells) from the previous iteration by splitting the most populous cells. Table 2 shows some of the codewords (at two different levels) used by MVQ to represent the Control Chart dataset (SYNDATA) [30].

Table 2. Codewords of a 2-level codebook that are used to represent SYNDATA in MVQ approximation.



3.2 Time Series Encoding

After a codebook is generated, we can form a new representation for each time series in the dataset. In the process of encoding, every series is decomposed into segments (i.e., subsequences) of length l (which is equal to the length of each codeword). For each segment, the closest (based on a distance metric) codeword in the codebook is then found and the corresponding index is used to represent this segment. After finding the corresponding codeword index for each segment, the appearance frequency of each codeword is counted.

The new representation of a time series is a vector $X'=f_1,f_2,\ldots,f_s$ showing the appearance frequency of every codeword. By applying this new encoding form, we can easily deal with time series with arbitrary large number of points, since we can always reduce their dimensionality to a rather small number given by the size, s, of the codebook.

3.3 Time Series Summarization

Besides achieving dimensionality reduction, this encoding process also provides a very nice summarization of the time series, which is useful in many applications. Table 2 shows different codewords we obtain using this method; these codewords stand for the most representative subsequences (of a given length) for the entire time series dataset. Instead of the whole time series, we may be more interested in the usage of representative key subsequences. This is very useful in the discovery of motifs or approximately repeated subsequences in time series [22]. In this case, we can just check the appearance frequencies of these codewords and get an overview of the time series. For example, in Figure 2, we show a time series representation using a number of codewords. Two of these codewords are being used twice revealing a pattern that would remain undetected using previous

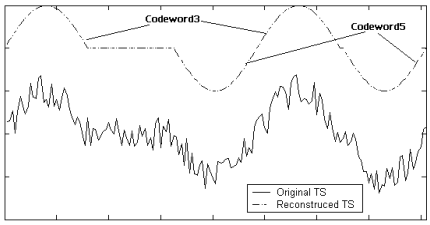


Figure 2. A time series (bottom) is being represented as a sequence of representative subsequences i.e., codewords (top). Two codewords (#3 and #5) are being used twice in this representation.

techniques. Results on time series summarization are presented in Section 4.4.

3.4 Distance measure and a multiresolution representation

Based on the frequency of appearance of key sequences within time series, features of time series are extracted forming a new representation of a rather small dimensionality and similarity retrieval can be efficiently performed. We still need a distance measure appropriate for this new representation. We choose the Histogram Model as the distance measure, and all the experimental results presented in Section 4 are based on it. By applying the histogram model, it is not difficult to identify the time series that are similar to a given query (i.e., that have similar frequent patterns). However, using only one codebook (analysis at a single resolution), introduces some problems that cannot be ignored.

First, although the local information of a time series is kept after the encoding process, the new representation of a time series is not recording the order among the indices of different codewords. Some important global information of the time series is lost in this representation. In Figure 3, we see two different time series whose encoded representations are the same (2, 1). This problem in the key subsequence representation correspondingly increases the number of false alarms reducing the performance of the single resolution (i.e., single codebook) method. On the other hand, in real applications, it is not always easy to find a suitable resolution (correspondingly, a suitable codeword length). Moreover, an inappropriate codeword length may reduce the efficiency.

In order to solve these potential problems occurring due to the use of a single resolution, we introduce a hierarchical mechanism, which involves several different resolutions for encoding. While the encoding form of higher resolution pays more attention to the detail of local information, that of lower resolution represents more global information. The piecewise approximation with VQ encoding is applied at several resolutions. For each resolution this is done by grouping a different number of consecutive segments together, i.e., the length of the

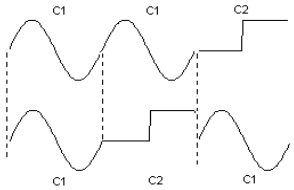


Figure 3. Necessity of multiresolution representation: different series with the same encoded representation.

segment at a given resolution is a multiple (usually double) the length of the segment at the immediate higher resolution representation. Thus, we call this representation Multiresolution Vector Quantized (MVQ) approximation.

Figure 4 shows a time series and its reconstruction series using different resolutions. (For different resolution levels, the sizes of codebooks are the same, 32, and the lengths of codewords are 128, 64, 32, 16, respectively.) By assigning reasonable weights to different resolutions, we define a new weighted similarity measure, the Hierarchical Histogram Model:

$$S_{HHM}(q,d_j) = \sum_{i=1}^{c} w_i * S_{HMi}(q, d_j) \qquad (2)$$

where c is the number of resolution levels.

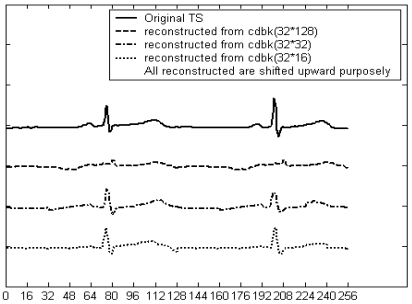


Figure 4. Reconstruction of time series using different resolutions

3.5 Parameters of MVQ

Here we discuss in more detail the parameters of MVQ and how to choose their values. For the number c of resolution levels an intuitive choice is $c = log\ n$, with the length of a codeword at the i^{th} level being 2^{i-1} ($1 \leq i \leq log\ n$). However, when the codeword is too short (e.g., of length 1, 2), this becomes meaningless. Thus, we need to set a minimum value of codeword length l_{min} and set the number of hierarchical levels as $c = \log (n / l_{min}) + 1$.

The codeword length (l) for each level is chosen as follows: At the first level, each time series is treated as a whole ($l = n$); at the second level, each time series is partitioned into two parts ($l = n/2$), and at the i^{th} level, $l = n / 2^{i-1}$. In cases where n is not a power of two we satisfy this constraint approximately.

The size of the codebook at each resolution level is data dependent, since the more subsequences used during the training process and the higher their variability, the

larger the size of the codebook needed. In fact, the higher the number of partitions and the number of codewords the better the approximation but also the more computation and space is needed. So, there is a tradeoff between efficiency and accuracy of approximation. In practice (as also shown in our experiments (see Section 4)), use of a rather small codebook can achieve very good results. In addition to the number of codewords, the Lloyd algorithm uses a threshold to stop the iterations when the fractional drop of the distortion between consecutive iterations reaches a certain point. A common value for this threshold is 0.01.

Our experiments show that a multiresolution representation achieves much higher accuracy than a single resolution one. The price for this improvement is slightly more computation, since we have to calculate the similarity at each resolution level before we can finally compute S_{HHM}. In our experiments we studied the behavior of the multiresolution approach with different weights assigned to each resolution level. Lacking any information or any prior knowledge about the domain (i.e., the most realistic case) the straightforward solution is to use equal weights for all resolutions. This choice provides the best results in almost all of the experiments we performed. The proposed method also provides the ability to include prior domain knowledge in the selection of the weights.

4. Experiments

In time series similarity analysis, best matches retrieval and clustering are two of the most common and important applications. We performed experiments to evaluate the effectiveness and efficiency of our method in these two applications. We address the following issues: (a) how accurate the method is, (b) how it compares to alternatives, (c) how fast and scalable it is. We start with a description of the datasets we used in our experiments.

4.1 Datasets

In the experiments presented in this section, one synthetic and two real datasets are involved. We used the Control Chart synthetic dataset (SYNDATA) which is downloadable from the UCI KDD archive [30]. This dataset contains 600 examples of control charts (each has 60 points) synthetically generated by the process in Alcock and Manolopoulos [3]. The time series belong to six different classes of control charts: Normal, Cyclic, Increasing trend, Decreasing trend, Upward shift, and Downward shift, with each class having 100 time series.

The first real dataset, CAMMOUSE, is a spatiotemporal dataset of 5 words obtained using the Camera Mouse Program [5]. The 2D time series obtained represent the X and Y position of a human tracking feature (e.g., tip of finger). In conjunction with a "spelling program" the user can "write" various words and the transitions of the tracking feature or word image's profiles are being recorded. We used 3 recordings of 5 words. The 5 words were: "Athens", "Berlin", "Boston", "London", and "Paris". For simplicity, only the x-values are considered. The average length of sequences in this dataset is 1100 points. The shortest one is 834 points and the longest one is 1719 points. Since the length of sequences varies for different instances, we stretched all sequences to a same length of 1600 points.

The second real dataset, RTT, consists of RTT (packet round trip time) measurements from UCR to CMU with sending rate of 50 msec for a day (Feb 10, 2002, starting at 8:20pm). The total number of RTT values is 1,728,000. The dataset was partitioned into 24 time series of length 72,000, each standing for an hour of RTT measurements. These measurements vary between 70 and 150. For clustering experiments we separated the time series into the following three classes based on the ratio of time where the RTT value is greater than 100: (a) heavy traffic hours: ratio > 0.5 (6 series), (b) medium traffic hours: 0.5 > ratio > 0.1 (7 series) and (c) light traffic hours: ratio < 0.1 (11 series).

In order to avoid the effects of scaling and shifting in the analysis, before we actually perform any experiment, we preprocess the datasets with zero-mean normalization. That is, each time series X is normalized as:

$$X = (X - \overline{X}) / \sigma(X)$$

where $\overline{X}$ is the mean value of X and $\sigma(X)$ is its standard deviation. For the RTT dataset we take logarithms before we apply the normalization.

4.2 Best Match Searching

4.2.1 Experiment design

The best match searching is defined as follows: given a query sequence, find the best k matches in the database (i.e., having the lowest dissimilarity with the query) or find all the time series whose dissimilarity with the query is below some predefined threshold. In order to evaluate the performance of different approaches in best match searching, we need an evaluation metric.

Definition 3. For a given query, the set of time series which are actually within the same class as the query (given our prior knowledge) is taken as the standard set (*std_set(q)*), and the results found by different approaches (*knn(q)*) are compared with this set. The matching accuracy is defined as:

$$\text{Accuracy} = \frac{|\,\text{knn(q)} \cap \text{std\_set(q)}\,|}{\text{k}} \times 100\% \quad (3)$$

In the definition above, *knn(q)*, is the *k* nearest neighbors for the query found by a certain method. In our experiments, every time series in the dataset is treated as a query, and the best *k* matches (*k* nearest neighbors) are sought within the whole dataset. The average accuracy of a certain method is then calculated based on the matching

Table 3. Experiment parameters for SYNDATA

	MVQ Parameters	
Level	*l*	*s*
1	60	6
2	30	16
3	20	16
4	10	32
5	5	32

results taking each time series as a query. The actual value of *k* we use depends on the number of time series within the same class. In our experiments, the value of *k* can vary, but for the purpose of demonstration, we just show the results when *k* is set to the number of time series within the same class.

4.2.2 Experiments on SYNDATA

In this section, we show the results of the experiments performed on the SYNDATA dataset. The experimental parameters for different resolution levels are given in Table 3. With the increase of resolution, the codeword length decreases and the size of codebook increases (since there are more training samples available for that resolution).

The experimental results on SYNDATA are shown in Table 4. The first element in the weight vector represents the weight assigned to the first level, the second element the weight assigned to the second level, and so on (e.g., with a weight vector [1 0 0 0 0], only the first level is involved in distance calculations). Accuracy is defined based on Eq. (3). The experimental results clearly demonstrate the effect of using a multiresolution approach: the combination of multiple resolutions dramatically improves the matching accuracy over the single resolution approach.

Table 4. Matching accuracy on SYNDATA

Method	Weight Vector	Accuracy
Single level VQ	[1 0 0 0 0]	0.55
	[0 1 0 0 0]	0.70
	[0 0 1 0 0]	0.65
	[0 0 0 1 0]	0.48
	[0 0 0 0 1]	0.46
MVQ	[1 1 1 1 1]	**0.83**
Euclidean		0.51

To show the effectiveness of the proposed representation and distance metric, we applied the plain Euclidean distance (naïve method) on the same dataset, which directly computes the Euclidean distance to measure the similarity between time series. From Table 4 we can conclude that for this dataset, the Naïve method does worse than most of the single level VQ approximations, while MVQ provides a much better matching accuracy.

Table 5. Experiment parameters for CAMMOUSE data

	MVQ parameters	
Level	*l*	*s*
1	1600	6
2	800	8
3	400	8
4	200	16
5	100	16

4.2.3 Experiments on CAMMOUSE

We performed similar experiments as with SYNDATA dataset. The experiment parameters and results are shown in Table 5 and 6 respectively.

Table 6. Matching accuracy on CAMMOUSE data

Method	Weight Vector	Accuracy
Single level VQ	[1 0 0 0 0]	0.56
	[0 1 0 0 0]	0.60
	[0 0 1 0 0]	0.44
	[0 0 0 1 0]	0.56
	[0 0 0 0 1]	0.60
MVQ	[1 1 1 1 1]	**0.83**
Euclidean		0.58

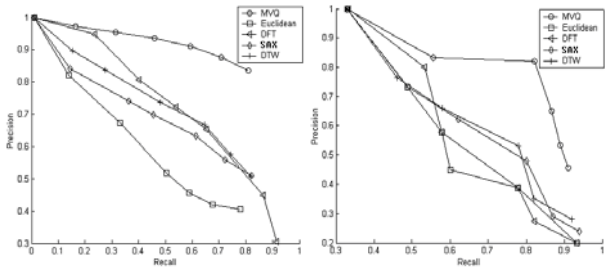
From Table 6, it is clear that for the CAMMOUSE dataset, the hierarchical mechanism also helps to improve the accuracy obtained with a single resolution level. Comparing with the average matching accuracy of the plain Euclidean method, the retrieval accuracy of MVQ is much better (25% higher).

4.2.4 Comparison with other methods

In order to compare the efficiency and accuracy of MVQ in similarity searches we considered alternative methods including the Discrete Fourier Transform (DFT), plain Euclidean, Dynamic Time Warping (DTW) and Symbolic Aggregate approXimation (SAX) [23]. For evaluation and comparison, every time series in the dataset is taken as a query, and the precision and recall pairs corresponding to the top *1,2,3,…,k* retrieved time series are calculated. Then the average value of precision and recall is computed for the whole dataset. The actual value for *k* is different for different methods.

For DFT, SAX and MVQ, some parameters need to be set up for the experiments. For DFT, we take the first 16 non-zero coefficients; for SAX the number of segments is set to 15 (SYNDATA) or 16 (CAMMOUSE) and the codebook size is set to 16. For MVQ we take the same codebook sizes as in previous subsection for the 5 resolution levels and use [1 1 1 1 1] as the weight vector.

Figure 5 shows the precision-recall performance on SYNDATA and CAMMOUSE. Notice that for a fixed recall ratio, the fewer time series are retrieved the better, and subsequently the higher the precision is. For both



(a) (b)

Figure 5. Precision-recall for different methods (a) on SYNDATA (b) on CAMMOUSE

datasets the precision decreases quickly with Plain Euclidean, DFT, SAX and DTW, while the precision with MVQ stays at a high level. MVQ achieves the best performance on these datasets. When the time series are short (as in the case of SYNDATA) MVQ's need for more space due to the multiple codebooks is noticeable. However, MVQ is the best distance function and provides the best accuracy. An interesting observation is that in most cases, even with only one layer, our distance measure can provide comparable or even better results than the other methods. Later in the experiments, we restrict the space requirements of MVQ so that they are comparable to those of the other methods.

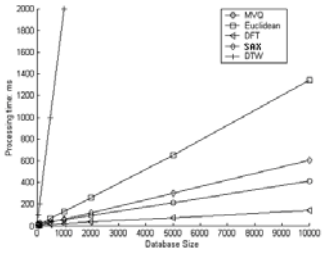


Figure 6. Processing time and scalability

Besides accuracy, other considerations for a good method should include speed and scalability. Figure 6 shows the processing time of different methods on datasets with various sizes. The experimental settings for different methods are the same as before. DFT shows the best processing efficiency with the shortest time, but considering the poor accuracy result shown in Figure5, it should not be taken as a good candidate.

In comparison to the other methods we considered here, although the encoding of the query consumes some time, MVQ outperforms them all in speed when the database size is not too small. Notice that the time reported here for MVQ does not include the preprocessing needed during the training phase to obtain the codebook (s) for a dataset. A brief discussion about the preprocessing cost can be found in Appendix B.

4.3 Clustering experiments

4.3.1 Experiment design.

For time series clustering, we conducted experiments on both synthetic and real datasets. The PAM (Partitioning Around Medoids) clustering algorithm was used to cluster the original time-series in every dataset. However, different approaches applied for distance calculation resulted in different distance matrices for the time series, and subsequently in different clustering results.

Table 7. Clustering accuracy of MVQ on SYNDATA

Method	Weight Vector	Accuracy
Single level VQ	[1 0 0 0 0]	0.69
	[0 1 0 0 0]	0.71
	[0 0 1 0 0]	0.63
	[0 0 0 1 0]	0.51
	[0 0 0 0 1]	0.49
MVQ	[1 1 1 1 1]	**0.82**
DFT		0.67
SAX		0.65
DTW		0.80
Euclidean		0.55

In order to evaluate the clustering accuracy and quality of our approach, a cluster similarity metric was used. Given two clusterings, $G=G_1,G_2, \ldots,G_k$ (the true clusters), and $A=A_1,A_2,\ldots,A_k$ (clustering result by a certain method), the clustering accuracy is evaluated with the cluster similarity defined as:

$$\mathrm{Sim(G,A)} = \frac{\sum_i \max_j Sim(G_i, A_j)}{k} \qquad (4)$$

where $\mathrm{Sim(G_i, A_j)} = \frac{2|\mathrm{G_i} \cap \mathrm{A_j}|}{|\mathrm{G_i}| + |\mathrm{A_j}|}$.

This metric was introduced in [11] to evaluate clustering results and was also used in [17]. The metric value ranges between 0 and 1, and it takes the maximal, i.e. 1, when the clustering result is perfect. For each dataset, we used the same experiment parameters as in Section 4.1. Considering the stochastic nature of the PAM algorithm, given a set of parameters, each experiment was repeated 10 times, and the average result is reported here. For the purpose of comparison, clustering results with other methods are also provided.

4.3.2 Experiments on SYNDATA dataset.

Taking the same parameters as shown in Table 3, clustering experiments were performed on the SYNDATA dataset. The experimental results are listed in Table 7. Clustering performance of other methods is also reported.

It is clear that for this dataset, we cannot achieve satisfying performance using the Euclidean Distance as the distance metric, while the suggested method is very promising. The performance achieved by several single resolution levels of the VQ approximation is better than that of the Naïve method (Euclidean on the original time series) and comparable or better to that of the other

Table 8. Clustering accuracy of MVQ on CAMMOUSE

Method	Weight Vector	Accuracy
Single level VQ	[1 0 0 0 0]	0.61
	[0 1 0 0 0]	0.60
	[0 0 1 0 0]	0.59
	[0 0 0 1 0]	0.63
	[0 0 0 0 1]	0.62
MVQ	[1 1 1 1 1]	**0.79**
DFT		0.62
SAX		0.58
DTW		0.69
Euclidean		0.61

methods. By combining different resolution levels, the clustering result is further improved.

For completeness we compared a multiresolution implementation of SAX to MVQ. We used 5 resolution levels with the number of segments as 2, 3, 6, 30 and 60 respectively. The accuracies of SAX with different resolutions vary between 0.54 and 0.65. However, when we tried to combine the distance measurement in all resolution levels, the accuracy was 0.64. Since SAX encodes already the order of segments in the original time series, the use of multiresolution levels does not improve the accuracy of the representation and its performance.

4.3.3 Experiments on CAMMOUSE dataset.

The experimental parameters for the CAMMOUSE dataset are the same as in Table 5. Table 8 displays the results for MVQ with different weight vectors and results of the other methods. Again, the performance of plain Euclidean Distance is poor, while MVQ provides much better clustering results. Its performance is also superior to the other methods we tested. Observe again that even with only one layer, our distance measure can provide comparative or even better results than the others (in this case MVQ has similar space requirements as the other methods).

4.3.4 Experiments on the RTT dataset.

For MVQ we used 5 different layers 1-5 with 3, 8, 8, 16, and 16 codewords respectively. This is a total of 51 codewords. We used the same number of parameters for DFT and SAX. Table 9 compares the clustering accuracy of MVQ with that of the other methods.

An important observation here is that we do not need to take all layers into consideration to get the best performance. The reason is that when the different resolution levels cannot present uniformly rich information, the involvement of less informative levels will reduce the overall accuracy. Furthermore, the study at different single resolution levels can help us identify the importance of different layers in discriminating among classes.

Table 9. Clustering results on RTT (with same space requirements for MVQ as for the other methods)

Method	Weight Vector	Accuracy
Single level VQ	[10000]	0.55
	[01000]	0.52
	[00100]	0.57
	[00010]	0.80
	[00001]	0.79
MVQ	[00011]	**0.81**
	[11111]	0.60
DFT		0.54
SAX		0.54
DTW		0.62
Euclidean		0.50

4.4 Summarizing time series

Here, we present results from applying MVQ to summarize time series. We consider the SYNDATA dataset. To help in evaluating the summarization capabilities of the proposed approach, in Figure 7, we present a few typical time series that we manually extracted from each of the six classes.

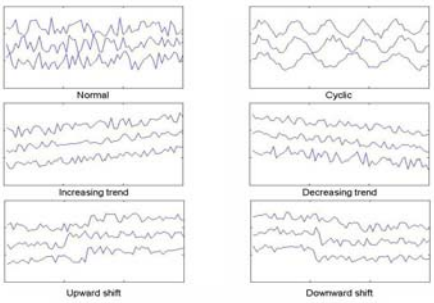


Figure 7. Representative time series extracted manually from the SYNDATA dataset.

Table 10 shows how the codewords of the first codebook are used to represent each class at the first level (of resolution). The actual codewords are displayed in Figure 8. The first number in each cell of Table 10 shows how usage of a codeword (row) is distributed across classes (we show percentages). These numbers add up to 100 for each row (codeword). The second number in each cell shows the usage (in percentages) of all codewords for a certain class (column). They add up to 100 for each column. One can make the following observations about the representation of classes at this level (more coarse approximation). For all time series in class 1 (normal) only the 2nd codeword is used and only class 2 (cyclic) time series use the same codeword (rarely though). The 2nd codeword is indeed very representative of the time series in class 1. Time series in class 2 make equal use of codewords 1, 5, and 6 while they rarely use codeword 2. Since class 2 is the cyclic one this makes a lot of sense. One could have a concise representation by just looking

Table 10. The codewords (*c*:1-6) used to represent each one of the 6 classes of SYNDATA at level 1.

c	class1	class2	class3	class4	class5	class6
1	0, 0	100,31	0, 0	0, 0	0, 0	0, 0
2	96,100	4, 4	0, 0	0, 0	0, 0	0, 0
3	0, 0	0, 0	0, 0	50,100	0, 0	50,100
4	0, 0	0, 0	50,100	0, 0	50,100	0, 0
5	0, 0	100,40	0, 0	0, 0	0, 0	0, 0
6	0, 0	100,25	0, 0	0, 0	0, 0	0, 0

at the codewords and the frequency of their use in different classes. Classes 3 (increasing trend) and 5 (upward shift) make equal use of the 4th codeword although no other classes use this codeword. They both have an increasing trend so this summarizes them very well. Similarly for classes 4 (decreasing trend) and 6 (downward shift) the 3rd codeword is used and no other class is using this codeword. At this first level we cannot discriminate between classes 3 and 5 and classes 4 and 6.

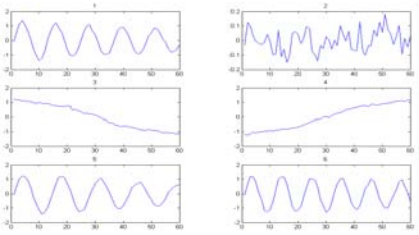


Figure 8. The codewords used to represent time series of the SYNDATA dataset at the first level.

The second level though provides more details into the summarization enabling the discrimination between classes 3 and 5 and classes 4 and 6. Table 11 shows how the codewords of the second codebook are used to represent each class at the second level. The actual codewords are displayed in Figure 9. Please note that

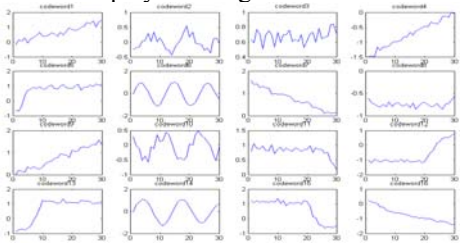


Figure 9. The codewords used to represent time series of the SYNDATA dataset at the second level.

codeword numbers correspond to different codewords (not the same codewords as for level 1). Time series in class 5 make heavy use of codeword 12 that indeed represents the upward shift. This is not the case for class 3 which instead uses heavily codewords 1 and 9. Similarly, class 6 makes heavy use of codeword 15 that indeed represents the downward shift. Class 4 uses the codeword 15 very rarely. The tables for the other levels are not shown here due to space limitations. They are also

Table 11. The codewords (c:1-16) used to represent each one of the 6 classes of SYNDATA at level 2.

c	class1	class2	class3	class4	class5	class6
1	2, 1	0, 0	94, 19	0, 0	2, 1	2, 1
2	90, 51	10, 6	0, 0	0, 0	0, 0	0, 0
3	1, 1	0, 0	0, 0	3, 1	56, 21	40, 15
4	0, 0	0, 0	96, 48	0, 0	3, 1	1, 1
5	0, 0	0, 0	37, 5	0, 0	63, 9	0, 0
6	11, 5	89, 39	0, 0	0, 0	0, 0	0, 0
7	0, 0	0, 0	0, 0	100, 45	0, 0	0, 0
8	0, 0	0, 0	1, 1	3, 2	46, 28	5, 29
9	0, 0	0, 0	98, 26	0, 0	2, 1	0, 0
10	78, 39	22, 11	0, 0	0, 0	0, 0	0, 0
11	0, 0	0, 0	0, 0	12, 3	25, 7	63, 19
12	0, 0	0, 0	7, 1	0, 0	93, 21	0, 0
13	0, 0	0, 0	0, 0	0, 0	100, 11	0, 0
14	4, 2	96, 44	0, 0	0, 0	0, 0	0, 0
15	0, 0	0, 0	0, 0	3, 1	0, 0	97, 15
16	1, 1	0, 0	0, 0	70, 48	0, 0	29, 20

not very useful for summarization of this particular dataset since most of the useful summarization information is extracted from the first two levels. These results demonstrate the ability of MVQ to provide a summarization of time series datasets. This is possible due to the symbolic and multiresolution nature of the representation.

5. Discussion

The MVQ approach that we proposed for representing time series data in order to make their analysis more efficient is a natural extension of the piecewise constant approximation schemes proposed earlier. By applying Vector Quantization to extract high-level features of the data and by involving a multiresolution approach we were able to identify a "vocabulary" of subsequences of various lengths and improve performance and efficiency in time series similarity retrieval. We were especially successful in domains where we could not achieve good results using the Euclidean distance as the similarity metric. In addition, the new representation is very useful in summarizing time series by providing typical patterns observed at different resolutions.

We presented the main idea of an approach to represent time series along with a new distance function that is better than previous distance functions and in addition it is fast to compute. Obviously, there are a lot of variations of this approach including use of sliding windows, non-rigid borders for subsequences, use of different rules for assigning weights to different resolutions, etc. These are directions in which this work can be extended. Another interesting problem is related to the size of the codebook. When we generate the codebooks for different resolutions, the size of each codebook affects the performance of encoding. The more codewords at a given resolution, the better the approximation but the efficiency of the method decreases.

Future studies include looking into these tradeoffs in more detail.

6. Conclusions

In this paper we introduced a new symbolic representation of time series, MVQ, along with a new distance function that is better than major competitors. By partitioning a sequence into equal-length segments and using vector quantization to represent each sequence by appearance frequencies of key subsequences, MVQ provides a more meaningful similarity metric for many domains, besides the improvement in efficiency because of the dimensionality reduction especially in the case of long sequences. Moreover, using a multiresolution approach, MVQ can record both local and global information of time series, which further improves the robustness in calculating similarity, requiring little more calculation than a single resolution approach.

The experimental evaluation of the proposed method showed that it outperforms current state-of-the-art methods in clustering and similarity searches. This is due to the following: (a) it exploits prior knowledge about the data, (b) it takes multiple resolutions into account and (c) it partially ignores the ordering of the 'codewords' within the time sequence due to the histogram model that it uses.

The proposed representation is symbolic potentially allowing the application of text-based retrieval techniques into the similarity analysis of time series. Moreover, due to the symbolic and multiresolution representation the proposed approach is excellent in summarizing time series by providing typical patterns observed at different resolutions. The proposed transformation on time series is very fast to process long time series, since the length of new representation is only related to the size of the codebook. The parameters of our method are easy to determine. In particular, a general conclusion from our experiments is that lacking any prior knowledge equal weights to all resolution levels works well most of the time. While the experimental results presented here mainly focus on similarity analysis, clustering, and summarization, our approach can also be easily adjusted to other applications, such as frequent pattern retrieval (i.e., motif discovery), association rule mining, and other data mining applications.

Acknowledgements

The authors are grateful to the anonymous referees and to Eamonn Keogh for providing helpful comments. This work was supported in part by NSF under Grant No. IIS-0237921, by NIH under Grant No. R01MH68066-01A1 (funded by NIMH, NINDS, and NIA) and by the Pennsylvania Department of Health.

References

[1] Agrawal, R., Faloutsos, C. and Swami, A.. "Efficient similarity search in sequence databases", Proceedings of the 4th Int'l Conference on Foundations of Data Organization and Algorithms. Chicago, IL, Oct 13-15, 1993. pp. 69-84.

[2] Agrawal, R., Lin, K. I., Sawhney, H. S. and Shim, K., "Fast similarity search in the presence of noise, scaling, and translation in time-series databases", Proceedings of the 21st Int'l Conference on Very Large Databases. Zurich, Switzerland, Sept., 1995, pp. 490-501.

[3] Alcock R.J. and Manolopoulos Y.. "Time-Series Similarity Queries Employing a Feature-Based Approach" Proceedings of 7th Hellenic Conference on Informatics, Ioannina, Greece, Aug. 27-29, 1999, pp.III.1-9.

[4] Baeza-Yates, R.A. & Gonnet, GH.. "A fast algorithm on average for all-against-all sequence matching", Proceedings of the String Processing and Information Retrieval Symposium, 1999, pp. 16-23.

[5] Betke, M., Gips, J., and Fleming, P., "The Camera Mouse: Visual Tracking of Body Features to Provide Computer Access For People with Severe Disabilities." *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10:1, March 2002, pp. 1-10.

[6] Chen, L. and Ozsu, M.T., "Multi-scale histograms for answering queries over time series data", Proceedings of the 20th International Conference on Data Engineering, Boston, MA, 2004, p. 838.

[7] Deligiannakis A., Kotidis, Y., and Roussopoulos, N., "Compressing historical information in sensor networks", Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, Paris, France, June 2004, pp. 527-538.

[8] Goldin, D.Q. and Kanellakis, P.C. "On similarity queries for time-series data: Constraint specification and implementation", Proceedings of Constraint Programming, Marseilles, France, 1995.

[9] Faloutsos, C., Jagadish, H., Mendelzon, A. and Milo, T., "A signature technique for similarity-based queries", Proceedings of the Int'l Conference on Compression and Complexity of Sequences. Positano-Salerno, Italy, Jun 11-13, 1997.

[10] Faloutsos, C., Ranganathan, M. and Manolopoulos, Y., "Fast subsequence matching in time-series databases", Proceedings of the ACM SIGMOD Int'l Conference on Management of Data. Minneapolis, MN, May 25-27, 1994, pp. 419-429.

[11] Gavrilov, M., Anguelov, D., Indyk, P. and Motwani, R., "Mining the stock market: Which measure is best? ", Proceedings of the International Conference on Data Mining and Knowledge Discovery, 2000, pp. 487-496.

[12] Gersho, A. and Gray R. M., *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1992.

[13] Gusfield, D., *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1997.

[14] Hetland, M. L., "A survey of recent methods for efficient retrieval of similar time sequences", In Mark Last, Abraham Kandel, and Horst Bunke, editors, *Data Mining in Time Series Databases*, World Scientific, 2004.

[15] Höppner, F., "Discovery of temporal patterns – learning rules about the qualitative behavior of time series", Proceedings of the 5th European Conference on Principles and Practice of Knowledge Discovery in Databases, Freiburg, Germany, 2001, pp. 192-203.

[16] Huhtala, Y., Kärkkäinen, J. & Toivonen, H., "Mining for similarities in aligned time series using wavelets", *Data Mining and Knowledge Discovery: Theory, Tools, and Technology*, SPIE Proceedings Series, Vol. 3695. Orlando, FL, Apr., 1999, pp. 150-160.
[17] Kalpakis, K., Gara, D. and Puttagunta, V, "Distance Measures for Effective Clustering of ARIMA Time-Series", Proceedings of the 2001 IEEE International Conference on Data Mining, San Jose, CA, Nov 29-Dec 2, 2001, pp. 273-280
[18] Keogh, E., Chakrabarti, K., Pazzani, M. and Mehrotra, S., "Locally adaptive dimensionality reduction for indexing large time series databases", Proceedings of ACM SIGMOD Conference on Management of Data. Santa Barbara, CA, May 21-24, 2001, pp 151-162.
[19] Keogh, E., Chakrabarti, K., Pazzani, M. and Mehrotra, S., "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases", *Journal of Knowledge and Information Systems*, 2001
[20] Keogh, E. & Folias, T., The UCR Time Series Data Mining Archive. http://www.cs.ucr.edu/~eamonn/TSDMA/index.html. Riverside CA. University of California, Computer Science & Engineering Department.
[21] Keogh, E. and Pazzani, M., "A simple dimensionality reduction technique for fast similarity research in large time series databases", Proceedings of the Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining, Kyoto, Japan, 2000.
[22] Lin, J., Keogh, E., Patel, P. and Lonardi, S., "Finding motifs in time series", The 2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, July 23 - 26, 2002.
[23] Lin, J., Keogh, E., Lonardi, S. and Chiu, B., "A Symbolic Representation of Time Series, with Implications for Streaming Algorithms", Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, San Diego, CA. June 13, 2003.
[24] Linde, S., Buzo, A. and Gray, A., "An algorithm for vector quantizer design", *IEEE Transactions on Communications*, vol. 28, 1980, pp. 84-95.
[25] Lloyd, S. P., "Least squares quantization in PCM", *IEEE Transactions on Information Theory*, IT(28), 1982, pp. 127-135.
[26] Megalooikonomou, V., Li, G., Wang, Q., "A Dimensionality Reduction Technique for Efficient Similarity Analysis of Time Series Databases", Proceedings of the 13th ACM CIKM International Conference on Information and Knowledge Management, Washington, DC, Nov. 8-13, 2004, pp. 160-161.
[27] Park, S., Chu, W.W., Yoon, J. and Hsu, C., "Efficient search for similar subsequences of different lengths in sequence databases", Proceedings of the ICDE, 2000, pp. 23-32.
[28] Piotr Indyk, Nick Koudas, S. Muthukrishnan. "Identifying Representative Trends in Massive Time Series Data Sets Using Sketches", Proceedings of VLDB, 2000, pp 363-372.
[29] Rafiei, D., "On similarity-based queries for time series data", Proceedings of the 15th International Conference on Data Engineering (ICDE), Sydney, Australia, 1999, pp. 410-417.
[30] UCI KDD Archive. http://kdd.ics.uci.edu
[31] Wu, Y., Agrawal, D. and El Abbadi, A., "A comparison of DFT and DWT based similarity search in time-series databases", Proceedings of the 9th ACM CIKM Int'l Conference on Information and Knowledge Management. McLean, VA, Nov 6-11, 2000, pp. 488-495.
[32] Yi, B-K and Faloutsos, C., "Fast Time Sequence Indexing for Arbitrary Lp Norms", Proceedings of the VLDB, Cairo, Egypt, Sept, 2000.
[33] Zhu, L., Rao, A. and Zhang A., "Theory of Keyblock-based Image Retrieval", *ACM Transactions on Information Systems*, 20(2), 2002, pp. 224-257.

Appendix

A. Time series codeword representation – Other models of similarity

In VQ-based image retrieval [33], two other models that have been proposed are the Boolean Model (BM) and the Vector Model (VM). The Histogram Model we adopted in our methodology can be considered as special case of VM. For completeness we present these models in the context of time series analysis below:

- *Boolean model (BM):* computes the similarity of the Boolean models of the codeword representation of two time series using the following formula:

$$S_{BM}(q,t) = n_{11} * w_{11} + n_{00} * w_{00}$$

  where n_{11} is the number of identical indices and n_{00} is the number of indices of the code words that do not exist in both of the representations, while w_{11} and w_{00} are the weights assigned to these frequencies.
- *Vector Model (VM):* computes the similarity between the frequency-based representations of two time series using the following formula:

$$S_{vm}(q,t) = \frac{\sum_{i=1}^{s} f_{i,t} * f_{i,q}}{\sqrt{\sum_{i=1}^{s} f_{i,t}^{2}} * \sqrt{\sum_{i=1}^{s} f_{i,q}^{2}}}$$

  In the above formula, $f_{i,t}$, denotes the frequency of codeword i in time series t.

B. Preprocessing cost: Codebook generation

In MVQ a codebook needs to be generated for each one of the multiresolution levels using training data before the encoding can be performed. Let i be the number of iterations in the training process where i depends on the predefined threshold of the fractional drop of the distortion. During each iteration, every training vector is compared to every codeword. Since the size of codebook is s, and totally there are $N*w$ training vectors (N is the number of time series in the training set and w is the number of fragments at the highest resolution of a time series), and c the number of resolution levels, the time complexity of preprocessing of a single level is: $T(\text{training}) = O(c * N * w * s * i)$. This time complexity is not so prohibitive since training is done once during preprocessing and as we showed earlier the size of the codebook needs not be large to achieve very good approximation using MVQ. In the case that the data is modified over time there is no additional overhead if the distributions remain the same. In the case of a decreased codebook quality an incremental update of the codewords need to be considered.