# A PCA-based Similarity Measure for Multivariate Time Series[*]

Kiyoung Yang and Cyrus Shahabi
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781
[kiyoungy,shahabi]@usc.edu

## ABSTRACT

Multivariate time series (MTS) datasets are common in various multimedia, medical and financial applications. We propose a similarity measure for MTS datasets, *Eros* (*E*xtended F*ro*beniu*s* norm), which is based on Principal Component Analysis (PCA). *Eros* applies PCA to MTS datasets represented as matrices to generate principal components and associated eigenvalues. These principal components and eigenvalues are then used to compare the similarity between MTS matrices. Though *Eros* in itself does not satisfy the triangle inequality, without which existing multidimensional indexing structures may not be utilized, the lower and upper bounds to satisfy the triangle inequality are obtained. In order to show the validity of *Eros* for similarity search on MTS datasets, we performed several experiments on three datasets (2 real-world and 1 synthetic). The results show the superiority of our approaches as compared to the traditional similarity measures for MTS datasets, such as Euclidean Distance (ED), Dynamic Time Warping (DTW), Weighted Sum SVD (WSSVD) and PCA similarity factor ($S_{PCA}$) in precision/recall.

## Categories and Subject Descriptors

H.2.4 [**Systems**]: Multimedia databases
; G.3 [**PROBABILITY AND STATISTICS**]: Time series analysis, Multivariate statistics

## General Terms

Algorithms, Measurement, Performance, Design, Experimentation

## Keywords

Similarity Measure, Multivariate Time Series, Principal Component Analysis, Singular Value Decomposition, Nearest Neighbor Search

## 1. INTRODUCTION

A time series is a series of observations, $x_i(t)$; $[i = 1, \cdots, n; t = 1, \cdots, m]$, made sequentially through time where $i$ indexes the measurements made at each time point $t$ [37]. It is called a univariate time series when $n$ is equal to 1, and a multivariate time series (MTS) when $n$ is equal to, or greater than 2.

MTS datasets are common in various fields, such as in multimedia, medicine and finance. For example, in multimedia, Cybergloves used in the Human and Computer Interface applications have around 20 sensors, each of which generates 50∼100 values in a second [16, 31]. For gesture recognition and video sequence matching using computer vision, several features are extracted from each image continuously, which renders them MTSs [6, 2, 26]. In medicine, Electro Encephalogram (EEG) from 64 electrodes placed on the scalp are measured to examine the correlation of genetic predisposition to alcoholism [41]. Functional Magnetic Resonance Imaging (fMRI) from 696 voxels out of 4391 has been used to detect similarities in activation between voxels in [11].

Univariate time series have been broadly explored by many researchers. A time series is often regarded as a point in multidimensional space. For similarity search in a multidimensional space, the Euclidean distance is often employed. Because of the dimensionality curse, many approaches have tried to reduce the dimension while preserving most of the "energy" of the time series in the reduced dimension. For example, DFT (Discrete Fourier Transform) [1], DWT (Discrete Wavelet Transform) [25] and SVD (Singular Value Decomposition) [20] have been used to transform the original time series. After transformation, only the first few or the best few coefficients are chosen to represent the original time series. With the reduced dimensions, the time series can be indexed using multidimensional indexing techniques, such as R-tree [12] and its variants. For more details on univariate time series analysis, please refer to [1, 25, 20, 7, 39].

By contrast, MTS datasets have not been extensively explored for $k$ nearest neighbor ($k$NN) searches. Each MTS item is usually stored in an $m \times n$ matrix, where $m$ is the number of observations and $n$ is the number of variables (e.g., sensors). An MTS item should be treated as a whole, since there are usually important correlations among the

[*]This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC), IIS-0238560 (CAREER) and IIS-0307908, and unrestricted cash gifts from Microsoft. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

*MMDB'04,* November 13, 2004, Washington, DC, USA.
Copyright 2004 ACM 1-58113-975-6/04/0011 ...$5.00.

variables in MTS datasets. These correlations will be lost if an MTS item is broken into multiple univariate time series and each processed separately and then aggregated to generate the result. For the same reason, an MTS item may not be transformed into one long univariate time series. With some domain knowledge as in [41], the variables can be organized into subsets of variables, and then the similarity can be checked using a subset of variables. However, the domain knowledge is not always available, and to find the interdependencies among the variables is in itself another challenge [37, 14]. One of the example queries we are interested in is, given a Cyberglove MTS data, find the most similar behavior performed by other users in a virtual reality environment.

In this paper, we propose a similarity measure *Eros* (*E*xtended F*ro*beniu*s* norm) for *k*NN searches in MTS databases. *Eros* is based on the Frobenius norm that is used to compute the matrix norm [22]. *Eros* extends the Frobenius norm to measure the similarity between two matrices using the principal components, i.e., the eigenvectors from the covariance matrices and eigenvalues. That is, instead of computing the similarities between the variables from two MTS items and aggregating up the result to produce the final similarity, we apply the following process. First, covariance matrices of the two MTS items are computed. Next, the eigenvectors and eigenvalues of the covariance matrices are calculated. Finally, the similarities between the corresponding eigenvectors from each MTS item are measured with weights that are based on the eigenvalues obtained from the MTS dataset. Though *Eros* in itself does not satisfy the triangle inequality, we can obtain the lower and upper bounds using the weighted Euclidean distance, which satisfies the triangle inequality. Without this property, the filter and refinement phases of existing multidimensional indexing techniques cannot be utilized, because false dismissals may occur in the filter phase [3, 5, 17].

We conducted several experiments on two real-world datasets: AUSLAN [16] obtained from UCI KDD repository [13] and the Human Gait dataset [36]; and one synthetic dataset: TRACE with 16 classes [28], and compared our approach with traditional MTS similarity measures in recall/precision. As shown in Section 5.3, *Eros* outperforms the other similarity measures, such as Euclidean distance (ED), Dynamic Time Warping (DTW), Weighted Sum SVD (WSSVD) [32] and Principal Component Analysis (PCA) similarity factor ($S_{PCA}$) [33, 21].

The remainder of this paper is organized as follows. Section 2 discusses the background. Our proposed methods are described in Section 3. This is followed by the experiments and results in Section 5. In Section 6, the related work is discussed. Conclusions and future work are presented in Section 7.

# 2. BACKGROUND

In this section, we briefly describe the distance metrics, SVD, PCA Similarity Factor and the Frobenius norm, from which our proposed method is extended. For more details on these topics, please refer to [20, 21, 22].

## 2.1 Distance Metrics

A distance metric, $D$, must have the following properties for all items $A$, $B$ and $C$[8].

- $D(A, B) \geq 0$
- $D(A, B) = 0 \iff A = B$
- $D(A, B) = D(B, A)$
- $D(A, B) + D(B, C) \geq D(A, C)$ (*triangle inequality*)

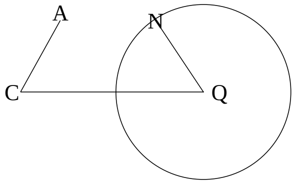


**Figure 1: Triangle Inequality.**

The examples of the distance metrics that satisfy the above properties are the Euclidean distance and the Manhattan distance. The fourth property ensures that the lower bound can be calculated. For example, as shown in Figure 1, if we know that $D(Q, N)$, where $Q$ is the query item and $N$ is the current nearest neighbor, is less than the lower bound of $D(Q, A)$, i.e., $D(C, A) - D(Q, C)$, then we do not need to compute $D(Q, A)$. The triangle inequality is used in indexing techniques for filter & refinement phases guaranteeing no false dismissals.

Since we use the weighted Euclidean distance to bound *Eros*, we describe the Euclidean distance and the weighted Euclidean distance for $n$ dimensional row normal vectors in more detail. Let $a$ and $b$ be $n$ dimensional row normal vectors, whose norms are all 1. The Euclidean distance between $a$ and $b$ is

$$D(a, b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} = \sqrt{2 - 2 < a, b >}$$

where $< a, b >$ is the inner product of $a$ and $b$. Let $w$ be a weight that is to be applied to the vectors and is greater than 0. The weighted Euclidean distance between $a$ and $b$ is

$$D_w(a, b, w) = \sqrt{\sum_{i=1}^{n} w(a_i - b_i)^2} = \sqrt{2w - 2w < a, b >}$$

In Section 3.3, we extend this to MTS data to determine the lower and upper bounds of *Eros*.

## 2.2 Singular Value Decomposition (SVD)

*Definition 1.* Singular Value Decomposition. Let A be a general real $M \times N$ matrix. The singular value decomposition (SVD) of A is the factorization

$$A = U\Sigma V^T \qquad (1)$$

where U is a column-orthonormal N × r matrix, r is the rank of the matrix A, $\Sigma$ is a diagonal r × r matrix of the eigenvalues $\lambda_i$ of A, where $\lambda_1 \geq \cdots \geq \lambda_r \geq 0$ and V is a column-orthonormal M × r matrix [20].

The eigenvalues and the corresponding eigenvectors are sorted in non-increasing order. V is called the right eigenvector matrix, and U the left eigenvector matrix.

Note that the eigenvectors obtained by applying SVD to covariance matrix are the same as the principal components.

Throughout this paper, SVD is only applied to covariance matrices. Hence, the eigenvectors and the principal components are used interchangeably.

## 2.3 PCA Similarity Factor

PCA Similarity Factor, $S_{PCA}$, is defined between two matrices of the same number of columns, but not necessarily the same number of rows. $S_{PCA}$ firstly obtains the principal components for each matrix, and chooses the first $k$ principal components based on heuristics. For example, the first $k$ principal components whose variances represent 95% of the total variance are chosen. $S_{PCA}$ then computes the similarity between the first $k$ principal components.

*Definition 2.* The PCA Similarity Factor, $S_{PCA}$, between two matrices, $A$ and $B$, is defined as follows [21]:

$$S_{PCA}(A, B) = trace(LM^T ML^T) = \sum_{i=1}^{k} \sum_{j=1}^{k} \cos^2 \theta_{ij}$$

where $L$ and $M$ are the matrices that contain the first $k$ principal components of $A$ and $B$, respectively, $\theta_{ij}$ is the angle between the $i$th principal component of $A$ and the $j$th principal component of $B$.

The range of $S_{PCA}$ is between 0 and $k$. Intuitively, $S_{PCA}$ measures the similarity between two matrices by computing the squared cosine values between all the combinations of the first $k$ principal components from two matrices, while the Frobenius norm measures the cosine values between the corresponding principal components as described in the following section.

## 2.4 Frobenius Norm

The Frobenius norm is one of the matrix norms [22], which is also called the Euclidean norm. The Frobenius norm is easy to compute, for example, when comparing how similar two matrices A and B are, using $||A - B||_F$ [22].

*Definition 3.* The Frobenius norm of an $m \times n$ matrix A is defined as follows [22]:

$$||A||_F = (\sum_{i=1}^{m} \sum_{j=1}^{n} (a_{ij})^2)^{1/2} = (trace(A^T A))^{1/2}. \qquad (2)$$

Let $x_i$ be a column vector of size $m$. $A$ can then be represented as $A = [x_1, \cdots, x_n]$. Subsequently, the squared Frobenius norm of $A$ is computed as follows:

$$||A||_F^2 = trace(A^T A) = \sum_{i=1}^{n} < x_i, x_i >$$

where $< a, b >$ is the inner product of $a$ and $b$.

Now, consider two right eigenvector matrices, A and B of size $n \times n$, obtained by applying SVD to covariance matrices, which are represented as $A = [a_1, \cdots, a_n]$ and $B = [b_1, \cdots, b_n]$, respectively. The squared Frobenius norm of A-B is then computed as follows:

$$\begin{aligned} ||C||_F^2 = trace(C^T C) &= 2n - 2 \sum_{i=1}^{n} < a_i, b_i > \\ &= 2n - 2 \sum_{i=1}^{n} \cos \theta_i \end{aligned}$$

where $C = A - B = [a_1 - b_1, \cdots, a_n - b_n]$, $a_i$ and $b_i$ are orthonormal vectors, and $\cos \theta_i$ is the angle between $a_i$ and $b_i$. Intuitively, the Frobenius norm between two eigenvector matrices computes the angles between the corresponding eigenvectors and sums them up.

| Symbol | Definition |
|---|---|
| $\mathbf{A}$ | an $m \times n$ matrix representing an MTS item |
| $\mathbf{A}^T$ | the transpose of $\mathbf{A}$ |
| $\mathbf{M_A}$ | the covariance matrix of size $n \times n$ for $\mathbf{A}$ |
| $\mathbf{V_A}$ | the right eigenvector matrix of size $n \times n$ for $\mathbf{M_A}$<br>$\mathbf{V_A} = [\ a_1,\ a_2,\ \cdots,\ a_n\ ]$ |
| $\mathbf{\Sigma_A}$ | an $n \times n$ diagonal matrix that has all the eigenvalues for $\mathbf{M_A}$ obtained by SVD |
| $a_i$ | a column orthonormal eigenvector of size $n$ for $\mathbf{V_A}$ |
| $a_{ij}$ | $j$th value of $a_i$, i.e.,<br>a value at the $i$th column and the $j$th row of $\mathbf{A}$ |
| $a_{*j}$ | all the values at the $j$th row of $\mathbf{A}$ |
| $w$ | a weight vector of size $n$<br>$\sum_{i=1}^{r} w_i = 1, \forall i\ w_i \geq 0$ |

**Table 1: Notations used in this paper**

Let us define the weighted Frobenius form that gives different weight to each column:

*Definition 4.* The weighted Frobenius norm of an $m \times n$ matrix $A$ is defined as

$$||A||_{W,F} = (\sum_{i=1}^{m} w_{ii} \sum_{j=1}^{n} (a_{ij})^2)^{1/2} \qquad (3)$$

where $W$ is a symmetric diagonal positive semidefinite matrix, i.e., $w_{ij} = 0$ where $i \neq j$, $\sum_{i=1}^{n} w_{ii} = 1$ and $w_{ii} \geq 0$ for all $i$.

Subsequently, the weighted Frobenius norm between two right eigenvector matrices, $A$ and $B$, can be computed as follows:

$$\begin{aligned} ||C||_{W,F} &= (\sum_{i=1}^{n} w_{ii} \sum_{j=1}^{n} |c_{ji}|^2)^{1/2} \\ &= (2 - 2 \sum_{i=1}^{n} w_{ii} < a_i, b_i >)^{1/2} \qquad (4) \\ &= (2 - 2 \sum_{i=1}^{n} w_{ii} \cos \theta_i)^{1/2} \end{aligned}$$

where $C = A - B = [a_1 - b_1, \cdots, a_n - b_n]$, $a_i$ and $b_i$ are orthonormal vectors, and $\cos \theta_i$ is the angle between $a_i$ and $b_i$.

# 3. THE PROPOSED ALGORITHM

In this section, we propose a similarity measure for MTS datasets *Eros*, based on our observations from both $S_{PCA}$ and the Frobenius norm. Table 1 lists the notations used in the remainder of this paper, if not specified otherwise. Note that in this paper, the whole matching queries are considered, and the sub-sequence matching queries are part of our future works (See Section 7).

## 3.1 Eros : Extended Frobenius norm

We first formally define our proposed similarity measure, *Eros*. Next, we provide the intuitions behind it.

*Definition 5. Eros* (*E*xtended F*ro*beniu*s* norm). Let $\mathbf{A}$ and $\mathbf{B}$ be two MTS items of size $m_A \times n$ and $m_B \times n$, respectively. Let $\mathbf{V_A}$ and $\mathbf{V_B}$ be two right eigenvector matrices by applying SVD to the covariance matrices, $\mathbf{M_A}$ and $\mathbf{M_B}$, respectively. Let $\mathbf{V_A} = [a_1, \cdots, a_n]$ and $\mathbf{V_B} = [b_1, \cdots, b_n]$, where $a_i$ and $b_i$ are column orthonormal vectors of size $n$. The *Eros* similarity of $\mathbf{A}$ and $\mathbf{B}$ is then defined as

$$\begin{aligned} Eros(\mathbf{A}, \mathbf{B}, w) &= \sum_{i=1}^{n} w_i | < a_i, b_i > | && (5) \\ &= \sum_{i=1}^{n} w_i | \cos\theta_i | && (6) \end{aligned}$$

where $< a_i, b_i >$ is the inner product of $a_i$ and $b_i$, $w$ is a weight vector which is based on the eigenvalues of the MTS dataset (see Section 3.2), $\sum_{i=1}^{n} w_i = 1$ and $\cos\theta_i$ is the angle between $a_i$ and $b_i$[1]. The range of *Eros* is between 0 and 1, with 1 being the most similar.

We now discuss *Eros*. This algorithm measures the similarity between two MTSs using the right eigenvector matrices that contain the principal components and associated eigenvalues. Using the right eigenvector matrices for similarity computation has the following advantages:

- Same size for all the MTS data items : In general, MTS items of a given application will have the same number of variables $n$, i.e., sensors, but different number of observations $m$. Comparing two MTS items with different sizes is a challenge. The size of a right eigenvector matrix, however, is fixed at $n \times n$. Thusly, the problem of different lengths is resolved.

- Dimension reduction: For MTS items, the number of observations $m$, is usually far greater than the number of variables, $n$. Considering that the size of a right eigenvector matrix is $n \times n$, the size of the data to be dealt with is greatly reduced.

Intuitively, *Eros* measures the similarity between two MTS items by comparing how far the principal components are apart using the aggregated eigenvalues as weights taking into account the variance for each principal component. Note that *Eros* only considers the acute angle that the two corresponding axes (eigenvectors) generate. The eigenvectors represent the axes with the maximum variances, not the direction [15] (see Equation (6)). Therefore, as depicted in Figure 2, when the angle ($\alpha$) between the two corresponding eigenvectors is not acute, we take the absolute value of the inner product and compute the similarity between the two corresponding eigenvectors using the acute angle ($\beta$). More specifically, the inner product of two normal vectors, $a$ and $b$ in Figure 2, yields $cos\alpha$, while what we need is $cos\beta = cos(\pi - \alpha) = -cos\alpha$. Therefore, we take the absolute value of the inner product, so that $cos\alpha$ is computed when $\alpha \leq \pi/2$, while $-cos\alpha$ is computed when $\alpha > \pi/2$.

Recall that, in general, there are three types of transformation that should be considered for similarity measures, i.e., shift, scale and time warping. The similarity measures for time series should then be invariant to those transformations. For *Eros*, the covariance matrix is computed by first subtracting the mean for each dataset, which addresses the shift transformation. Moreover, the covariance matrix represents how scattered the data generated by one variable are in relation to the other variables by computing the covariances with all the other variables. Our intuition is that the scatteredness would be similar for the data with the same label. In addition, the size of covariance matrix is fixed at $n \times n$, where $n$ is the number of variables. These properties are empirically shown to address the time warping transformation in Section 5.

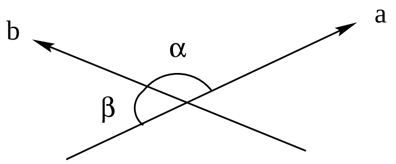


**Figure 2: Two corresponding eigenvectors, $a$ and $b$**

Intuitively, using the correlation matrix, not the covariance matrix, would have addressed the scale transformation, since the correlation is obtained by dividing the covariance with the standard deviations. However, the performance of *Eros* using the correlation matrix is worse than the one using the covariance matrix. This may suggest that for the datasets used in our experiments the scale should not be considered as invariant. That is, if the scale is different for two MTS items, then those two MTS items should be considered different.

## 3.2 Computing Weights

Recall that by applying SVD to the covariance matrices, we obtain not only the principal components but also the eigenvalues that represent the variances for principal components. When comparing two MTS items, *Eros* considers both the principal components and the eigenvalues. In this section, we propose two heuristics for computing the weight vector $w$ for *Eros* based on the eigenvalues obtained from the MTS dataset satisfying the following conditions:

- $\sum_{i}^{n} w_i = 1$
- $w_i \geq 0$ for all i

Note that the same weight vector should be used for all the similarity measure computations for each $k$NN search. Hence, the eigenvalues obtained from all the MTS items in the database are aggregated into one weight vector as in Algorithm 1 and 2. Algorithm 1 computes the weight vector $w$ based on the distribution of raw eigenvalues, while Algorithm 2 first normalizes each $s_i$, and then calls Algorithm 1. Function $f()$ in Line 3 of Algorithm 1 is an aggregating function, e.g., min, mean and max. Intuitively, each $w_i$ in the weight vector represents the aggregated variance for all the $i$th principal components. The weights are then normalized so that $\sum_{i=1}^{n} w_i = 1$.

We store one weight vector whose size is $n \times 1$ for the *Eros* approach, where $n$ is the number of variables. Additionally, we store the number of items in the database and the unnormalized weight vector, so that the weight vector can be updated when items are inserted into, or removed from the database.

[1] For simplicity, it is assumed that the covariance matrices are of full rank. In general, the summations in Equation (5) and (6) should be from 1 to min($r_A$, $r_B$), where $r_A$ is the rank of $\mathbf{M_A}$ and $r_B$ the rank of $\mathbf{M_B}$.

**Algorithm 1** Computing a weight vector $w$ based on the distribution of raw eigenvalues

```
function computeWeightRaw(S)
Require: an $n \times N$ matrix S, where n is the number of variables for the dataset and $N$ is the number of MTS items in the dataset. Each column vector $s_i$ in S represents all the eigenvalues for $i$th MTS item in the dataset. $s_{ij}$ is a value at column i and row j in S. $s_{*i}$ is $i$th row in S. $s_{i*}$ is $i$th column, i.e, $s_i$.
for i=1 to n do
    $w_i \leftarrow f(s_{*i})$;
end for
for i=1 to n do
    $w_i \leftarrow w_i / \sum_{j=1}^{n} w_j$;
end for
```

**Algorithm 2** Computing a weight vector $w$ based on the distribution of normalized eigenvalues

```
function computeWeightRatio(S)
Require: the same as Algorithm 1.
for i=1 to N do
    $s_i \leftarrow s_i / \sum_{j=1}^{n} s_{ij}$;
end for
computeWeightRaw(S);
```

## 3.3 Bounding Eros

We first define the *Eros distance metric*, $D_{Eros}$ which preserves the similarity relation of *Eros*. Subsequently, we describe two weighted Euclidean distances between two eigenvector matrices, which are used as the upper and lower bounds of $D_{Eros}$.

*Definition 6.* $D_{Eros}$ is defined as:

$$\begin{aligned} D_{Eros}(\mathbf{A}, \mathbf{B}, w) &= \sqrt{2 - 2\textstyle\sum_{i=1}^{n} w_i | < a_i, b_i > |} \\ &= \sqrt{2 - 2\textstyle\sum_{i=1}^{n} w_i | \sum_{j=1}^{n} a_{ij} \times b_{ij} |} \end{aligned} \tag{7}$$

$D_{Eros}$ preserves the similarity relation of *Eros*. That is, if **B** is more similar to **A** than to **C**, then the *Eros* similarity between **A** and **B** is greater than that between **B** and **C**, while the *Eros* distance between **A** and **B** is shorter than that between **B** and **C**.

*Lemma 1.* The similarity relation with *Eros* is reversely preserved with $D_{Eros}$.

PROOF. Assume that the following inequality with *Eros* holds.

$$Eros(\mathbf{A}, \mathbf{B}, w) > Eros(\mathbf{B}, \mathbf{C}, w)$$

which means

$$\sum_{i=1}^{n} w_i | < a_i, b_i > | \quad > \quad \sum_{i=1}^{n} w_i | < b_i, c_i > |$$

Therefore, we obtain the following:

$$D_{Eros}(\mathbf{A}, \mathbf{B}, w) = \sqrt{2 - 2\textstyle\sum_{i=1}^{n} w_i | < a_i, b_i > |}$$

$$< \sqrt{2 - 2\textstyle\sum_{i=1}^{n} w_i | < b_i, c_i > |} = D_{Eros}(\mathbf{B}, \mathbf{C}, w)$$

Consequently, the similarity relation with *Eros* is reversely preserved with $D_{Eros}$. □

Now, we use weighted Euclidean distance to bound $D_{Eros}$. Let us consider the weighted Euclidean distance, called $D_{max}$, between **A** and **B**:

$$\begin{aligned} D_{max}(\mathbf{A}, \mathbf{B}, w) &= \sqrt{\textstyle\sum_{i=1}^{n} \sum_{j=1}^{n} w_i (a_{ij} - b_{ij})^2} \\ &= \sqrt{2 - 2\textstyle\sum_{i=1}^{n} w_i \sum_{j=1}^{n} a_{ij} b_{ij}} \end{aligned} \tag{8}$$

The distance measure $D_{max}$, which is a weighted Euclidean Distance metric, satisfies the triangle inequality. Let us consider one more distance metric.

$$\begin{aligned} D_{min}(\mathbf{A}, \mathbf{B}, w) &= \sqrt{\textstyle\sum_{i=1}^{n} \sum_{j=1}^{n} w_i (|a_{ij}| - |b_{ij}|)^2} \\ &= \sqrt{2 - 2\textstyle\sum_{i=1}^{n} w_i \sum_{j=1}^{n} |a_{ij} b_{ij}|} \end{aligned} \tag{9}$$

$D_{min}$ is hence defined by computing $D_{max}$ between two MTS items **A** and **B**, after obtaining the absolute values of $a_i$ and $b_i$. Similarly, $D_{min}$ satisfies the triangle inequality. The upper and lower bounds of $D_{Eros}$ are then obtained as in the following lemma.

*Lemma 2.* The upper and lower bounds of $D_{Eros}$ are $D_{max}$ and $D_{min}$, respectively.

PROOF. Consider the summation parts in $D_{Eros}$, $D_{max}$ and $D_{min}$ in Equations (7), (8) and (9), respectively. Thus, we can find the inequalities among them as follows :

$$\sum_{i=1}^{n} w_i \sum_{j=1}^{n} |a_{ij} b_{ij}| \geq \sum_{i=1}^{n} w_i | \sum_{j=1}^{n} a_{ij} b_{ij} | \geq \sum_{i=1}^{n} w_i \sum_{j=1}^{n} a_{ij} b_{ij}$$

Hence, we conclude

$$D_{min} \leq D_{Eros} \leq D_{max} \tag{10}$$

□

Consequently, we derived the upper and lower bounds of $D_{Eros}$, i.e., $D_{min}$ and $D_{max}$, that satisfy the triangle inequality.

# 4. THE OVERALL PROCESS

In this section, we describe how to preprocess the MTS dataset and then how to perform the $k$NN search using $D_{Eros}$.

## 4.1 Pre-processing

Given MTS items stored in the database, the pre-processing is done according to Algorithm 3. Firstly, for all the MTS items, the eigenvalues and the right eigenvector matrices are computed by performing SVD on the covariance matrices. Using Algorithms 1 or 2, the weight vector $w$ is computed. Recall that the weight vector $w$, can change when items are added to or removed from the database.

**Algorithm 3** Preprocessing Algorithm of *Eros*

```
Require: the number of all the MTS items in the dataset $N$.
for i=1 to $N$ do
    A ← the $i$th MTS item in the database;
    B ← covariance matrix of A;
    [C, D, E] ← SVD(B);
    $s_i$ ← the eigenvalues in D;
    store E as the $i$th right eigenvector matrix;
end for
Compute the weight vector $w$ using Algorithm 1 or 2.
```

Recall that the time complexity of SVD for an $n \times n$ matrix is $O(n^3)$ by optimized batch algorithms, such as the one used in $MATLAB^{TM}$ [4, 10]. However, the time complexity of SVD is not a concern; it was shown to be acceptable with a dataset that contains 70,000 time series of length 492 for clustering [35]. On a PC with a 800MHz Pentium III CPU, a $64 \times 64$ matrix A can be decomposed in less than 0.04 second with one line MATLAB code, [a,b,c]=svd(A). Note also that MTS items in the database are preprocessed with SVD offline. The MTS query is then the only item that requires SVD during the online process, whose cost is negligible as compared to that of the $k$NN search.

### 4.2 $k$NN Search

Once the MTS datasets are preprocessed for *Eros*, we perform $k$NN searches. We describe $k$NN search using two-phase sequential scan in Algorithm 4. Note that though sequential scan is performed, still the two phased, filter and refinement scheme is utilized. Hence, firstly the data items are filtered out using the lower bound of $D_{Eros}$, i.e., $D_{min}$ between the query item and the MTS items in the database in Line 6. This filter phase can be performed efficiently using an indexing structure such as Spatial Transform Technique (STT) in [30], of which the distance metric is the quadratic form distance function $d_M^2(p,q) = (p-q)M(p-q)^T$, where $M$ is positive definite (i.e, $d_M^2(p,q) > 0$) which may change from the user relevance feedback without the need to rebuild the index. Due to the high dimensionality of MTS items, the distance-based indexing techniques, such as iDistance [40], would perform better for MTS datasets. To devise a distance-based indexing structure for *Eros* is part of our future work. In the refinement phase, $D_{Eros}$ is computed in Line 8. When a new NN is found, the *nndist* and *nnid* arrays are updated in Line 12, so that the elements in the *nndist* array are sorted in non-decreasing order of *Eros* distance.

**Algorithm 4** $k$NN Search Algorithm of *Eros* (Two-phase Sequential Scan)

```
Require: Given an MTS user query $Q$ and a weight vector $w$
 1: for i=1 to k do
 2:   nndist[i] ← $\infty$;
 3: end for
 4: for i=1 to N do
 5:   P ← the eigenvector matrix for the $i$th MTS item in the database;
 6:   res ← $D_{min}(P,Q,w)$;
 7:   if res $\leq$ nndist[k] then
 8:     res ← $D_{Eros}(P,Q,w)$;
 9:     if res $\leq$ nndist[k] then
10:       nndist[k] ← res;
11:       nnid[k] ← $i$;
12:       update nndist, nnid;
13:     end if
14:   end if
15: end for
```

In order to validate our proposed similarity measure *Eros*, a modified leave-one-out $k$NN search as in Algorithm 5 is used in Section 5. That is, for each query item, we check how many items were retrieved from the database in order to retrieve $r$ relevant items, where $1 \leq r \leq maxr$. Relevant items are the items in the database that have the same labels as the query item. Subsequently, for each $r$, the precision is computed (Line 12). Finally, we compute the average precisions across all the items in the database for each $r$ (Lines 18-20).

**Algorithm 5** Modified Leave-One-Out $k$ Nearest Neighbor Search for Recall-Precision graph

```
Require: the number of MTS items in the dataset, $N$, $k$, the maximum number of relevant items, $maxr$;
 1: for i=1 to 10 do
 2:   precision[i] ← 0;
 3: end for
 4: for i=1 to N do
 5:   Q ← $i$th item in the dataset;
 6:   $k \leftarrow 1$;
 7:   $r \leftarrow 1$;
 8:   repeat
 9:     Perform kNN search for Q;
10:     c ← the number of the same label items as Q in the k items retrieved;
11:     if $c = r$ then
12:       precision[r] ← precision[r] + c / k;
13:       r ← r + 1;
14:     end if
15:     $k \leftarrow k + 1$;
16:   until r $\geq$ $maxr$;
17: end for
18: for i=1 to 10 do
19:   precision[i] ← precision[i] / N;
20: end for
```

## 5. PERFORMANCE EVALUATION

### 5.1 Datasets

The experiments have been conducted on two different real-world datasets, AUSLAN and HumanGait and one synthetic dataset TRACE16, which are all labeled MTS datasets whose labels are given.

The Australian Sign Language (AUSLAN) dataset [16] uses 22 sensors on the hands to gather the datasets generated by signing of a native AUSLAN speaker. It contains 95 distinct signs, each of which has 27 examples. In total, the number of signs gathered is 2565. The size of the right eigenvector is 22 × 22 and the average length is around 60.

The Human Gait dataset from [36] has been used for identifying a person by gait recognition at a distance. In order to capture the gait data, a twelve-camera VICON system was utilized with 22 reflective markers attached to each subject. For each reflective marker, 3D position, i.e., x,y and z, are acquired at 120Hz, generating 66 values at each timestamp. 15 subjects, which are the labels assigned to the dataset, participated in the experiments and were required to walk at four different speeds, nine times for each speed. The total number of data items is 540 (15 × 4 × 9) and the average length is 133.

The Transient Classification Benchmark (TRACE) datasets have been used in [28] for plant diagnostics. For the TRACE dataset with 16 classes (TRACE16), each class has 100 examples. There are 5 variables, out of which the first 4 variables are the four signals and the 5th is the class label. The change of the class label from 0 to a class number (from 1 to 16) means the start of the transient. Using this information, we first located exactly where the transient starts and ends and removed those signals with class label 0. The size of the right eigenvector is 4 × 4 and the average length is 250. Note that in [28], the focus is to find the transition over the continuous data stream where the starting point is

|  | AUSLAN | TRACE16 | Human Gait |
|---|---|---|---|
| # of variables | 22 | 4 | 66 |
| average length | 60 | 250 | 133 |
| # of labels | 95 | 16 | 15 |
| # of items per label | 27 | 100 | 36 |
| total # of items | 2565 | 1600 | 540 |

**Table 2: Summary of datasets used in the experiments.**

unknown, while the focus of this paper is to find $k$ nearest neighbors given a query MTS item, assuming that the endpoints of all MTS items are accurately located. Finding $k$ nearest neighbors over the continuous data stream is part of our future research directions (see Section 7).

Table 2 shows the summary of the datasets used in the experiments. Therefore, for the AUSLAN dataset, for example, the Euclidean distance and dynamic time warping should deal with MTS items of size 60 × 22, while *Eros*, WSSVD deal with MTS items of size 22 × 22. This dimension reduction naturally results in reduced elapsed time as shown in Section 5.3.

Note that though labeled datasets, where each data is given a label, are used in order to validate our proposed similarity measures, the classification is not of primary interests to us, which could have been accomplished by model-based machine learning approaches, such as Hidden Markov Model (HMM) or Support Vector Machines (SVM). In most cases where the MTSs are generated, a class label may not be given to each data item, since there is no clear definition for classes. For example, in a virtual reality environment, human behavior is captured by devices such as magnetic trackers and gloves. However, each behavior may not be given a class label, such as frustration and being lost, since there are no clear definitions for them yet. In addition, MTSs are continuously generated, which would mean that the size of training datasets keeps increasing. Consequently, the models for HMM and SVM should be re-generated frequently, which would make model based approaches less attractive.

## 5.2 Methods

In order to validate our proposed similarity measure *Eros*, we performed modified leave-one-out $k$NN search as in Algorithm 5. For simplicity, we chose 10 for $maxr$. Recall that each dataset used in the experiments has more than 10 relevant items as shown in Table 2. For example, AUSLAN has 95 labels and each label has 27 items. The recall-precision graph [9] is then plotted, which has been frequently used to measure the performance of Content Based Image Retrieval (CBIR) systems [34, 19] as well as Information Retrieval (IR) systems.

For *Eros*, the weight vector $w$ is computed using both Algorithm 1 and 2 excluding the eigenvalues of the query item. We employ three different aggregating functions, i.e., mean, min and max. Subsequently, the one with the best performance will be presented for *Eros*. We then compare the performance of *Eros* with those of 4 other distance measures, i.e., the Euclidean Distance (ED), Dynamic Time Warping (DTW), Principal Component Analysis (PCA) similarity factor ($S_{PCA}$) and Weighted Sum SVD (WSSVD).

DTW is a technique for performing time-alignment and has been extensively employed in various applications, such as speech recognition [29] and time series similarity search [17]. Though DTW can be applied to 2 MTS items regardless of the items' lengths, the performance is shown to be the best when the ratio of the items' lengths is close to 1 [23], and the indexing technique for DTW is available only when the two items are of the same length [17, 27]. Also, ED is not defined for 2 MTS items with different lengths. Hence, before applying ED and DTW, all the MTS items are linearly interpolated to be of identical length. We chose this length to be the average length for each dataset as in [23]. For DTW, the $MATLAB^{TM}$ DTW code in [22] is used with slight modifications. A global limit on the maximum amount of warping Q is set to 10% of the length, and the distance between points, i.e., the local distance, is modified to be the square of the Euclidean distance, as in [17]. In addition, for ED and DTW, the experiments were performed with and without z-normalization, which renders the data zero mean and unit variance. The better performance between the two are then presented. The ED and DTW with z-normalization are denoted as EDZ and DTWZ, respectively.

$S_{PCA}$ is a similarity measure for MTS datasets [21, 33]. It first finds $k$ principal components (PCs), such that the corresponding $k$ eigenvalues describe more than, e.g., 95% of the total variance. Only the $k$ PCs are then used to compare the similarities between MTS items, and the eigenvalues are not utilized. For $S_{PCA}$ similarity measure, we tried both 95% and 99% of the total variance for each dataset, which are denoted as $S_{PCA}95$ and $S_{PCA}99$, respectively.

In [32], we proposed *WSSVD*, which employs the eigenvectors and eigenvalues of MTS items to compute the similarities between items using the inner product of the eigenvectors with the eigenvalues as weights. Note that the eigenvectors utilized in *WSSVD* are not principal components, since SVD is applied on the transpose-multiplication of the data matrix, not on the covariance matrix. We tried both transpose-multiplication as in [32] and covariance matrix for *WSSVD*, denoted as *WSSVD* and $WSSVD_{COV}$, respectively.

Note that $S_{PCA}$ and WSSVD cannot be used with existing indexing techniques because they do not satisfy the triangle inequality[2].

## 5.3 RESULTS

Table 3 shows the elapsed time to compute the similarity between two MTS items using 5 different similarity measures used in the experiments, i.e., *Eros*, $S_{PCA}99$, WSSVD, Euclidean distance (ED) and DTW. As expected, *Eros*, $S_{PCA}99$ and WSSVD take less time than ED and DTW, since the formers use the dimension reduced representations of MTS items.

Figure 3 shows that *Eros* gives the best recall-precision ratio for the AUSLAN dataset. Poor performances of ED and DTW may indicate that there are correlations among

[2] Consider three 2 × 2 column-orthonormal eigenvector matrices,

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \, and \; C = \begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{\sqrt{2}}{\sqrt{3}} \\ \frac{\sqrt{2}}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \end{pmatrix}$$

whose eigenvalues are [2 0.1]. The triangle inequality $\overline{AB} + \overline{BC} \geq \overline{AC}$ does not hold in $S_{PCA}$(when the first column eigenvector and the corresponding eigenvalue are used, since the first eigenvalue describe more than 95% of the total variance [33]) and WSSVD.

|  | AUSLAN | TRACE16 | Human Gait |
|---|---|---|---|
| Eros | 4.48E-05 | 3.13E-05 | 2.38E-04 |
| $S_{PCA}99$ | 1.01E-04 | 4.48E-05 | 1.18E-03 |
| WSSVD | 9.27E-05 | 6.98E-05 | 4.00E-04 |
| ED | 5.93E-04 | 2.17E-04 | 3.31E-03 |
| DTW | 9.21E-03 | 1.38E-01 | 2.45E-02 |

**Table 3: Elapsed time to compute the similarity between two MTS items in seconds.**

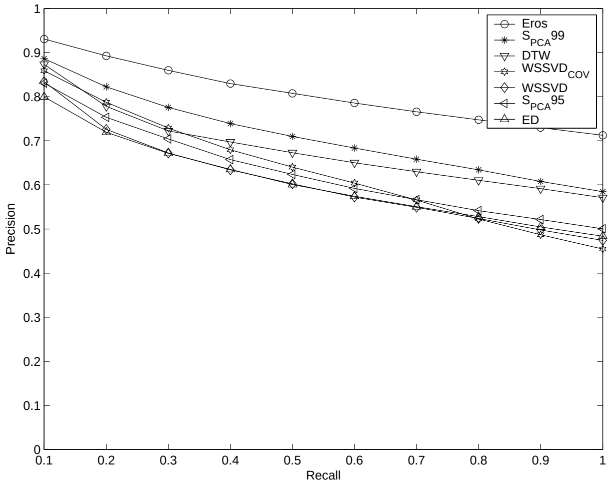


**Figure 3: Recall-precision graph (AUSLAN)**

variables, which are not considered by those two similarity measures. Note also that the performances of EDZ and DTWZ are worse than those of ED and DTW. This may suggest that for the AUSLAN dataset, the similarity measure should not be scale invariant. For *Eros*, the covariance matrices are used.

Figure 4(a) shows that for TRACE16, *Eros* and $WSSVD_{COV}$ perform similarly. However, $WSSVD_{COV}$ does not satisfy the triangle inequality, and hence cannot be used with an existing indexing technique, while *Eros* can. *Eros* outperforms all the other similarity measures. When recall is equal to 1.0, *Eros* outperforms ED, DTW, $S_{PCA}$ and WSSVD by more than 33%, 64% 69% and 100%, respectively. For TRACE16, ED and DTW again outperform EDZ and DTWZ.

Figure 4(b) represents similar results for the Human Gait dataset. *Eros* and $S_{PCA}$ yield similar performances. Note again that for *Eros*, an indexing structure can be utilized for efficient retrieval, while $S_{PCA}$ cannot. For Human Gait, EDZ and DTWZ perform much better than ED and DTW. This result may indicate that, before using ED and DTW, the dataset should be analyzed to determine whether z-normalization should be employed or not. Interestingly, however, *Eros* using the covariance matrix still performs better than the one using the correlation matrix. This may require further investigations.

Recall that intuitively, $S_{PCA}$ considers only the angles between the corresponding principal components (see Section 2.3), *WSSVD* in [32] and $WSSVD_{COV}$ consider both the angles between the corresponding principal components and the variance for each principal component and *Eros* considers both the *acute* angles and the variances for the principal components (see Section 3.1), when comparing two MTS items. Figure 3 shows that for the AUSLAN dataset, the acute angles and the variances should be considered. Figure 4(a) indicates that for the TRACE16 dataset, it does not matter whether the angles are acute or blunt, but the variances should be taken into account. Figure 4(b) represents that for the Human Gait dataset, the acute angles should be considered, but maybe not the variances. These results show that for $S_{PCA}$ and $WSSVD$, the characteristics of the dataset should be examined first, so that it can be decided whether $S_{PCA}$ or $WSSVD$ can be applied to the dataset. However, with *Eros* the prior examination of the dataset's characteristics may not be required, as shown in Figures 3 and 4.

## 6. RELATED WORK

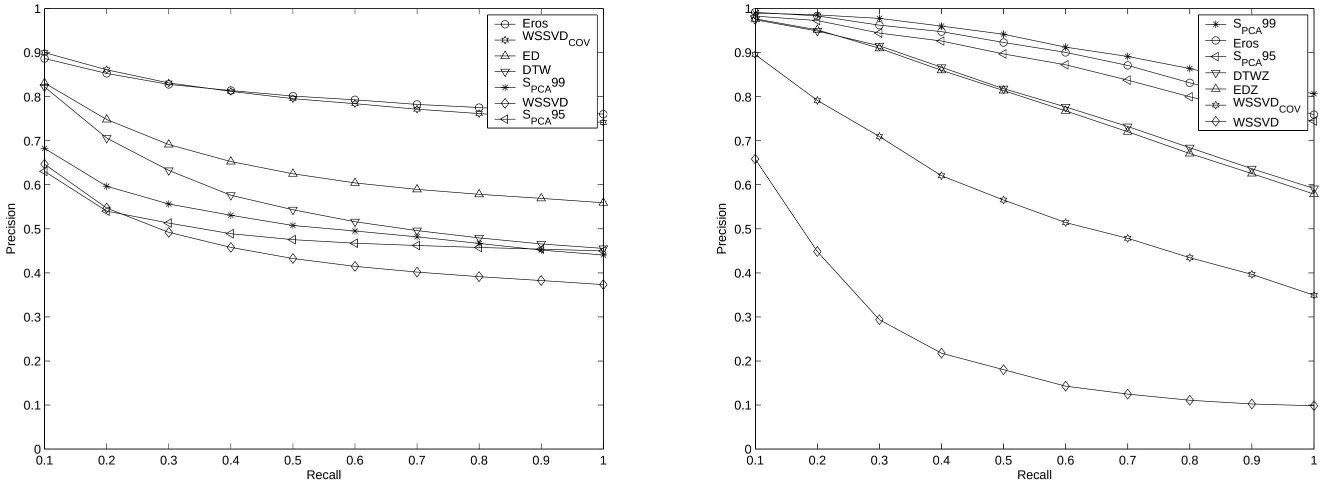
In [24], Oates proposes a method for identifying distinctive subsequences in MTS items, which is based on probabilistic models. Hence, they first try to build a model for each distinctive sequence, then they perform the distinctive subsequence matching based on the generated models. However, this model-based approach has the inherent scalability problem that the model should be re-generated when data are inserted or removed from the database [38]. In this paper, our focus is on the whole matching queries on the entire database. The subsequence matching queries will be part of our future research directions.

In [38], Vlachos *et.al* propose a technique for the retrieval of object trajectories, which can be regarded as an MTS item. However, they only deal with the datasets of two or three dimensional space for their similarity measure, Longest Common SubSequence (LCSS). In their experiments, they only used 2 attributes (X and Y) of the AUSLAN dataset, as opposed to the 22 attributes used in our experiments. Out of 95 distinct signs, only 10 signs with 5 samples for each sign are used for the experiments. Moreover, they used hierarchical clustering whose space complexity is approximately $O(N^2/2)$, which may not be suitable since the clustering is required whenever data are added to the training sets.

## 7. CONCLUSIONS AND FUTURE WORK

We proposed a similarity measure for MTS datasets, *Eros*, which is based on Principal Component Analysis. In order to compute the similarity between two MTS items, *Eros* compares the similarity between the corresponding principal components of the MTS items using the associated eigenvalues as weights. Experimentally, we show the validity of our proposed measure. In precision/recall, *Eros* outperforms ED, WSSVD, DTW and $S_{PCA}$ by a minimum of 6.5% when recall is 0.1 and as much as 100% when recall is 1. In elapsed time, *Eros* also benefits from using the dimension reduced representations of MTS items and takes less time to compute than the Euclidean distance and dynamic time warping.

We intend to extend this work in two directions. First, we plan to extend our technique to continuous data streams generating the result as soon as new data arrives. Second, in this paper, we considered only the whole matching queries. Subsequence matching queries for MTS datasets is one of the interesting future directions of this work. Instead of using sliding window-based approaches for subsequence matching, we plan to use a change detection technique, such as in [18], to segment the whole sequence into multiple sub-sequences. This way, the time complexity for



**(a) TRACE16**

**(b) Human Gait dataset**

**Figure 4: Recall-precision graphs**

the sub-sequence matching would be much lower than the sliding window-based approaches.

## 8. ACKNOWLEDGMENTS

The authors would like to thank Prof. Eamonn Keogh for providing us the dataset for the experiments and the anonymous reviewers for their very constructive comments.

## 9. REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient Similarity Search In Sequence Databases. In *FODO*, 1993.

[2] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic. Discovering clusters in motion time-series data. In *IEEE CVPR*, 2003.

[3] C. Bohm, S. Berchtold, and D. A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3), 2001.

[4] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *ECCV*, 2002.

[5] E. Chavez, G. Navarro, R. Baeza-Yates, and J. L. Marroqun. Searching in metric spaces. *ACM Computing Surveys*, 33(3), 2001.

[6] A. Corradini. Dynamic time warping for off-line recognition of a small gesture vocabulary. In *IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, 2001.

[7] G. Das, D. Gunopulos, and H. Mannila. Finding similar time series. In *PKDD*, 1997.

[8] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Interscience, second edition, 2001.

[9] W. B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, 1992.

[10] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins Univ Press, 1996.

[11] C. Goutte, P. Toft, E. Rostrup, F. A. Nielsen, and L. K. Hansen. On clustering fmri time series. *NeuroImage*, 9(3), 1999.

[12] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of SIGMOD*, 1984.

[13] S. Hettich and S. D. Bay. The UCI KDD Archive. *http://kdd.ics.uci.edu*, 1999.

[14] F. Hoppner. Learning dependencies in multivariate time series. In *Proc. of the ECAI'02 Workshop*, 2002.

[15] J. E. Jackson. *A User's Guide to Principal Components*. Wiley-Interscience, 1991.

[16] M. W. Kadous. *Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series*. PhD thesis, University of New South Wales, 2002.

[17] E. Keogh. Exact indexing of dynamic time warping. In *VLDB*, 2002.

[18] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Very Large Databases*, 2004.

[19] D.-H. Kim and C.-W. Chung. Qcluster: relevance feedback using adaptive clustering for content-based image retrieval. In *SIGMOD*, pages 599–610. ACM Press, 2003.

[20] F. Korn, H. V. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *SIGMOD*, 1997.

[21] W. Krzanowski. Between-groups comparison of principal components. *JASA*, 74(367), 1979.

[22] T. K. Moon and W. C. Stirling. *Mathematical Methods and Algorithms for Signal Processing*. Prentice Hall, 2000.

[23] C. Myers, L. R. Rabiner, and A. E. Rosenberg. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *IEEE TASSP*, ASSP-28(6), 1980.

[24] T. Oates. Identifying distinctive subsequences in

multivariate time series by clustering. In *KDD*, 1999.

[25] K. pong Chan and A. W.-C. Fu. Efficient time series matching by wavelets. In *ICDE*, 1999.

[26] C. Rao, A. Gritai, M. Shah, and T. Syeda-Mahmood. View-invariant alignment and matching of video sequences. In *IEEE ICCV*, 2003.

[27] T. M. Rath and R. Manmatha. Lower-bounding of dynamic time warping distances for multivariate time series. Technical Report MM-40, University of Massachusetts Amherst, 2002.

[28] D. Roverso. Plant diagnostics by transient classification: The aladdin approach. *IJIS,Special Issue on Intelligent Systems for Plant Surveillance and Diagnostics*, 2002.

[29] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE TASSP*, ASSP-26(1), 1978.

[30] Y. Sakurai, M. Yoshikawa, R. Kataoka, and S. Uemura. Similarity search for adaptive ellipsoid queries using spatial transformation. In *VLDB*, 2001.

[31] C. Shahabi. AIMS: An immersidata management system. In *VLDB CIDR*, 2003.

[32] C. Shahabi and D. Yan. Real-time pattern isolation and recognition over immersive sensor data streams. In *the 9th International Conference On Multi-Media Modeling*, 2003.

[33] D. Singhal, A.; Seborg. Clustering of multivariate time-series data. In *Proceedings of the American Control Conference*, volume 5, 2002.

[34] R. O. Stehling, M. A. Nascimento, and A. X. Falcão. A compact and efficient image retrieval approach based on border/interior pixel classification. In *CIKM*, pages 102–109. ACM Press, 2002.

[35] M. Steinbach, P.-N. Tan, V. Kumar, S. Klooster, and C. Potter. Discovery of climate indices using clustering. In *KDD*, 2003.

[36] Tanawongsuwan and Bobick. Performance analysis of time-distance gait parameters under different speeds. In *4th International Conference on Audio- and Video Based Biometric Person Authentication*, Guildford, UK, June 2003.

[37] A. Tucker, S. Swift, and X. Liu. Variable grouping in multivariate time series via correlation. *IEEE TSMC, Part B*, 31(2), 2001.

[38] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *ICDE*, 2002.

[39] Y.-L. Wu, D. Agrawal, and A. E. Abbadi. A comparison of DFT and DWT based similarity search in time-series databases. In *CIKM*, 2000.

[40] C. Yu, B. C. Ooi, K.-L. Tan, and H. V. Jagadish. Indexing the distance: An efficient method to KNN processing. In *The VLDB Journal*, 2001.

[41] X. L. Zhang, H. Begleiter, B. Porjesz, W. Wang, and A. Litke. Event related potentials during object recognition tasks. *Brain Research Bulletin*, 38(6), 1995.