In Proceedings of IEEE 16th International Conference on Data Engineering, San Diego, USA, 2000/Mar/1-3.

Similarity Search for Multidimensional Data Sequences*

Seok-Lyong Lee Seok-Ju Chun Deok-Hwan Kim Ju-Hong Lee Chin-Wan Chung
Korea Advanced Institute of Science and Technology, Taejon, Korea
{sllee, chunsj, dhkim, jhlee, chungcw}@islab.kaist.ac.kr

Abstract

Time-series data, which are a series of one-dimensional real numbers, have been studied in various database applications. In this paper, we extend the traditional similarity search methods on time-series data to support a multidimensional data sequence, such as a video stream. We investigate the problem of retrieving similar multidimensional data sequences from a large database. To prune irrelevant sequences in a database, we introduce correct and efficient similarity functions. Both data sequences and query sequences are partitioned into subsequences, and each of them is represented by a Minimum Bounding Rectangle (*MBR*). *The query processing is based upon these MBRs, instead of scanning data elements of entire sequences.*

Our method is designed (1) *to select candidate sequences in a database, and* (2) *to find the subsequences of a selected sequence, each of which falls under the given threshold. The latter is of special importance in the case of retrieving subsequences from large and complex sequences such as video. By using it, we do not need to browse the whole of the selected video stream, but just browse the substreams to find a scene we want. We have performed an extensive experiment on synthetic, as well as real data sequences* (*a collection of TV news, dramas, and documentary videos*) *to evaluate our proposed method. The experiment demonstrates that 73-94 percent of irrelevant sequences are pruned using the proposed method, resulting in 16-28 times faster response time compared with that of the sequential search.*

1. Introduction

In the last decade, time-series data became important in various database applications such as data mining or data warehousing. It includes a sequence of real numbers which represent values at time points, such as prices of stocks or commercial goods, weather patterns, sales indicators, biomedical measurements [12]. It is however basically a series of *one*-dimensional data, and thus the existing works until now have focused on indexing or searching *one*-dimensional sequences of data. As the use of multimedia data is widely spread in many application domains, the efficient retrieval of voluminous and complex information, which is the intrinsic characteristic of multimedia data, is getting increasingly important.

The problem we address in this paper is to design a scheme for searching a database to find multidimensional data sequences efficiently that are similar to a given query sequence. A multidimensional data sequence is a series of data elements, each element being represented by a multidimensional vector. Time-series data can be modeled by replacing each vector entry of a sequence with a single scalar value. Typical examples of a multidimensional data sequence include:

1) *video stream.* It consists of multiple frames and each frame is characterized by multiple feature attributes such as color, texture or shape. For example, a frame can be represented by a multidimensional vector in the RGB or YCbCr color space, by averaging color values of pixels of a frame or segmented blocks of a frame. The video stream is modeled as a trail of points in a multidimensional data space such that each frame of the stream constitutes a multidimensional vector (or point), whose components are feature values of a frame.

2) *image.* It can be represented as a multidimensional data sequence with spatial information considered. An image is segmented to a number of regions that can be ordered appropriately, based on space filling curves such as the Z-curve, gray coding, or the Hilbert curve [8]. This ordering forms a series of regions, each of which is represented by a vector of multiple feature values of a region.

Over a collection of sequences in a large database, the following are some examples of typical similarity queries:

One-dimensional sequence: Queries on traditional time-series data, such as *'Identify companies whose stock prices show similar movements during the last year to that of a given company,'* or *'Determine products with similar selling patterns to a given product'*

Multidimensional sequence: Queries on video streams or images, such as *'Find video streams that are similar to a given stream of news video,'* or *'Find all images in a database that contain regions similar to regions of a given image'*

The traditional query processing on time-series data

*This research was supported by the Basic Research Program (grant number 99-2-315-001-3) of the Korea Science and Engineering Foundation and the Software Technology Enhancement Program 2000 of the Ministry of Science and Technology of Korea.

uses the concept of a sliding window to get a smoother trail [5]. By using the window (size *w*) over a data sequence, a one-dimensional data sequence is converted to a *w* dimensional sequence. The *w* dimensional sequence is partitioned into subsequences, each of which is represented by a Minimum Bounding Rectangle (MBR) which tightly encloses all points in the subsequence. Its dimensionality is reduced in this process to avoid the '*dimensionality curse problem.*' The generated MBRs are indexed and stored into a database. On the other hand, a query sequence is divided into one or more subsequences of size *w*, each of which is represented by a *w* dimensional *point*. This query point acts as a representative of *w* values in the query sub-sequence. Query processing is based upon each query point and the MBRs of a data sequence stored in a database. However, the semantics of a point in *natural* multidimensional sequences such as video streams is different from that of one-dimensional time-series data. In the case of a multi-dimensional sequence, a point itself is a vector in a multi-dimensional space which holds various feature values. Traditional methods may not be applicable since we cannot map multiple multidimensional points of a query subsequence into one representative point, like time-series data. In this case, a query should be processed based upon every point in a query sequence, which causes a severe overhead. It is usual in video search that a key frame is selected for each shot, and a query is processed on the selected frames [6]. But the search by a key frame does not guarantee the correctness since it cannot always summarize all the frames of a shot. This is the motivation of our proposed method. It is designed to retrieve multidimensional sequences efficiently in a large database with the correctness preserved. A brief sketch of our proposed method is as follows:

Index construction. A multidimensional data sequence is generated from raw materials such as a video stream or an image by an appropriate parsing and dimensionality reduction process. It is partitioned into subsequences, and each of them is contained in an MBR which will be indexed and stored into a database for later processing, by using the R-tree [7] or its variants [2, 3, 4, 9].

Query processing. A query sequence is also divided into one or more MBRs upon which query processing is based. Two distance metrics are developed to handle the similarity search between MBRs and some pruning algorithms are proposed to minimize 'false hits' while guaranteeing the correctness of 'no false dismissal'. The method is designed, first, to select candidate sequences from a database, and next, to find the subsequences of a selected sequence, each of which falls under the given threshold.

In the case of retrieving subsequences from large and complex sequences such as video data streams, we believe the method can be of benefit. In video browsing, for example, we do not need to browse the whole stream of a selected video, but just browse the sub-streams found by the process. Another desirable property of our method is that it does not restrict the length of sequences. Both data sequences and query sequences are of arbitrary length. It is also allowed that a given query sequence may be longer than a data sequence, In this case, a query is processed to find data sequences to which the subsequences of the given query sequence are similar. In addition to the typical queries mentioned above, the following types of queries can be handled by the proposed method:

Query on finding sub-streams: *'Select videos in a database which contain the sub-streams that are similar to a given news video, and play those sub-streams only.'*

Long query (A query sequence is longer than a data sequence.): *'Find video streams in a database to which the sub-streams of a given video are similar.'*

The rest of the paper is organized as follows: Section 2 provides a survey of related works with a discussion of the similarity search for time-series data. Our proposed work is described in Section 3 including formal definitions of distance metrics used in this paper and a similarity search algorithm. Section 4 presents experimental results. We give conclusions in Section 5.

2. Related works

Several similarity search methods on time-series data have been proposed. Agrawal et al. [1] studied the whole sequence matching, in which the sequences to be compared are of the equal length. They introduced the Discrete Fourier Transform (DFT) to map time sequences to the frequency domain, to solve the dimensionality curse problem. Each sequence, whose dimensionality is reduced by using DFT, is mapped to a lower-dimensional point in the frequency domain, and is indexed and stored using the R* tree. This technique, however, has a restriction that a database sequence and a query sequence should be of equal length. Faloutsos et al. [5] proposed a fast subsequence matching technique that allows the similarity search between different-length sequences. They used a sliding window over a data sequence and extracted its features. A transformed data sequence is divided into subsequences, each of which is represented by an MBR. This MBR is indexed and stored using 'the *ST-index.*' Rafiei et al. [12] proposed a set of safe linear transformations of a given sequence that can be used as the basis for similarity queries on time-series data. They formulated operations such as moving average, reversing, and time warping. These transformations are extended to the multiple transformation in [10], where an index is searched only once and a collection of transformations are simultaneously applied to the index, instead of searching the index multiple times and each time applying a single transformation. Yi et al. [13] also addressed the time warping function which permits local accelerations and decelerations.

All above methods address the similarity search for one-dimensional time-series data, and thus do not handle

multidimensional data sequences. They also focus on selecting candidate sequences in a database which fall under the given threshold with a query sequence. To our knowledge, there has been no approach that is designed to handle the sophisticated similarity search, such as finding subsequences of a selected sequence as described in Section 1.

3. Proposed work

In this section, we discuss our proposed method in detail. Table 1 summarizes the notations used in this paper.

Table 1. Notations used in this paper

Symbol	Definition
S	Data sequence in a database
$S[i]$	The i-th entry of S
$S[i:j]$	Subsequence of S from entry i to j
Q	Query sequence
$mbr_k(S)$	The k-th MBR of S
ε	User-specified threshold
$d(*,*)$	Euclidean distance between two multidimensional points
$D(*,*)$	Distance between two sequences
$D_{mean}(*,*)$	Mean distance between two sequences of equal length
$D_{mbr}(*,*)$	Distance between two MBRs
$D_{norm}(*,*)$	Normalized distance between two MBRs

3.1. Nature of multidimensional data sequence

A multidimensional data sequence can be represented by a trail of points in a multidimensional data space. More formally, we describe it by the following definition:

Definition 1 A multidimensional data sequence S is defined as a series of its component vectors, $S = (S[1],S[2], \ldots,S[k])$, where each vector $S[k]$ is composed of multiple (n) scalar entries, that is, $S[k] = (S[k,1],S[k,2],\ldots,S[k,n])$.

We can formulate time-series data as a special case of this data model, by replacing each element $S[k]$ of S with a *one*-dimensional scalar component. Thus, it becomes a sequence of real numbers, each number representing a value at a time point.

The similarity of two objects, each of which is represented by a multidimensional vector, is generally defined as a function of the Euclidean distance (hereafter, referred as 'distance') between those two vectors. For example, images or video frames are represented as feature vectors, such as color, texture, or shape. The similarity between images or video frames can be described as a function of the distance between the corresponding feature vectors. The value range of the similarity between two objects is usually [0,1] while the range of the distance is [0, ∞]. The distance is close to zero when two objects are similar, and becomes large if they are quite different. But the similarity is the opposite. It is close to 1 when two objects are similar, while it is close to zero when they are very dissimilar. The distance between two objects can be transformed into the similarity by an appropriate mapping function. In this paper, a data space is normalized in the $[0,1]^n$ hyper-cube, where the length of each dimension is 1, and thus the maximum allowable distance is $\sqrt{n}$, a diagonal of the cube. This distance will be easily mapped to the similarity. We will use the distance for the similarity measure for simplicity. Now, let us consider the distance between two multidimensional data sequences, S_1 and S_2. The distance between two arbitrary points in each n-dimensional sequence S_1 and S_2, is given as:

$$d(S_1[i], S_2[j]) = \left(\sum_{1 \le t \le n} |S_1[i,t] - S_2[j,t]|^2\right)^{1/2}$$

where $S_1[i]$ and $S_2[j]$ are i-th and j-th points of sequences S_1 and S_2, respectively.

The distance between sequences, however, has the different semantics from that between points. In the case of multidimensional data sequences, each sequence is comprised of a number of multidimensional points. It is not suitable if we just use the sum of distances between all pairs of points in the two sequences to represent the distance between two sequences, since a pair of similar sequences with more points may produce the greater distance value than dissimilar sequences with fewer points. The following example shows this.

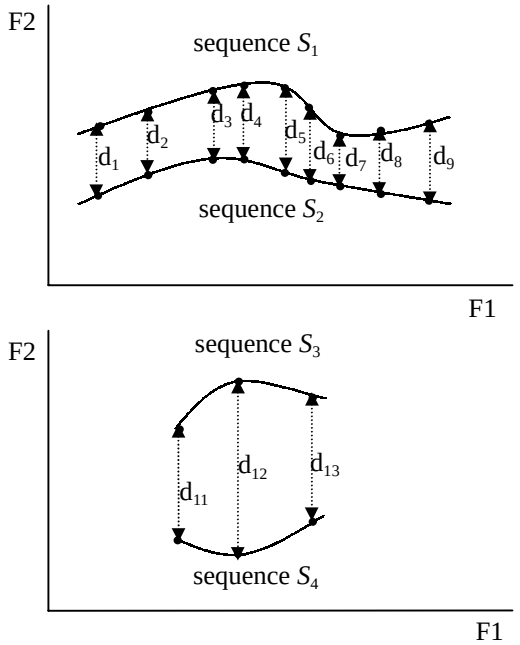

Figure 1. Distance between sequences

Example 1 Let us consider two pairs of sequences, (S_1, S_2) with 9 points each, and (S_3, S_4) with 3 points each. As we can see intuitively in Figure 1, the sequences S_1 and S_2 are apparently more similar than the sequences S_3 and S_4

since pairs of points of the former are placed closer than those of the latter. But, if we adopt the sum of distances as a similarity measure, we may get the result that the sequences S_3 and S_4 are closer (more similar). It shows that the sum of distances is not appropriate for the similarity measure of sequences. ■

The first metric we introduce is to measure the distance between two sequences. First, it is defined for same-length sequences, and then, extended to different-length sequences.

Definition 2 The distance $D(S_1,S_2)$ between two multi-dimensional sequences S_1 and S_2 of equal length, each of which has k points, is defined as the mean distance of the two, where the mean distance is defined as follows:

$$D(S_1, S_2) = D_{mean}(S_1, S_2) = \frac{\sum_{1 \le i \le k} d(S_1[i], S_2[i])}{k}$$

Next, let us consider the different-length sequences which cannot be compared directly, point by point. In this case, the shorter sequence is compared with the other by sliding from the beginning to the end. The shortest distance of each pair is adopted as the distance between two sequences. The following definition describes it more formally.

Definition 3 The distance $D(S_1,S_2)$ between two multi-dimensional sequences S_1 and S_2 of different length, each of which has k and m points respectively (Without loss of generality, we assume $k \le m$.), is defined as the minimum mean distance of every pair, where the minimum mean distance is defined as follows:

$$D(S_1, S_2) = \min_{1 \le j \le m-k+1} D_{mean}(S_1[1:k], S_2[j:j+k-1])$$

3.2. Distance metrics between MBRs

To measure the distance between MBRs, we introduce the MBR distance D_{mbr} between two MBRs. Generally, an MBR M in the n-dimensional Euclidean space is represented by two endpoints L(low point) and H(high point) of its major diagonal [11]. It will be:

$$M = (L, H)$$

where $L=(l_1,l_2,..,l_n)$, $H=(h_1,h_2,..,h_n)$, $l_i \le h_i$ for $1 \le i \le n$.

The distance between two MBRs in the 2-dimensional space is depicted in Figure 2, depending on their relative placement.

It is straightforward that the distance for the 2-dimensional space is extended to the n-dimensional space. Thus the following definition holds.

Definition 4 The MBR distance D_{mbr} between two MBRs $A=(L_A, H_A)$ and $B=(L_B, H_B)$, in the n-dimensional Euclidean space, is defined as the minimum Euclidean distance between two hyper-rectangles. That is:

$$D_{mbr}(A, B) = \left(\sum_{1 \le k \le n} x_k^{\,2} \right)^{1/2},$$

$$\text{where,} \quad x_k = \begin{cases} |h_{A,k} - l_{B,k}| & \text{if } h_{A,k} < l_{B,k}; \\ |l_{A,k} - h_{B,k}| & \text{if } h_{B,k} < l_{A,k}; \\ 0 & \text{otherwise .} \end{cases}$$

Observation 1 The distance D_{mbr} is shorter than the distance between any pair of points, one in a sequence S_1 and the other in a sequence S_2. That is:

$$D_{mbr}(A, B) \le \min_{a \in S_1, b \in S_2} d(a, b)$$

where A and B are the MBRs containing S_1 and S_2, respectively.

Lemma 1 (Lower Bounding Distance D_{mbr})
The shortest MBR distance D_{mbr} between any pair of MBRs in a query sequence Q and a data sequence S is the *lower bound* of the distance $D(Q,S)$ of two sequences. When N, R are the sets of MBRs in Q, S respectively, the following holds:

$$\min_{i \in N, j \in R} D_{mbr}(mbr_i(Q), mbr_j(S)) \le D(Q, S)$$

Proof: Let Q, S have k, l points respectively. Without loss of generality, we can assume $k \le l$.

$$D(Q, S) = \min_{1 \le u \le l-k+1} D_{mean}(Q[1:k], S[u:u+k-1])$$

$$= \min_{1 \le u \le l-k+1} \left(\sum_{1 \le v \le k} d(Q[v], S[u+v-1]) \Big/ k \right)$$

Let δ be the shortest distance between arbitrary two points, one in a sequence Q and the other in a sequence of S, then

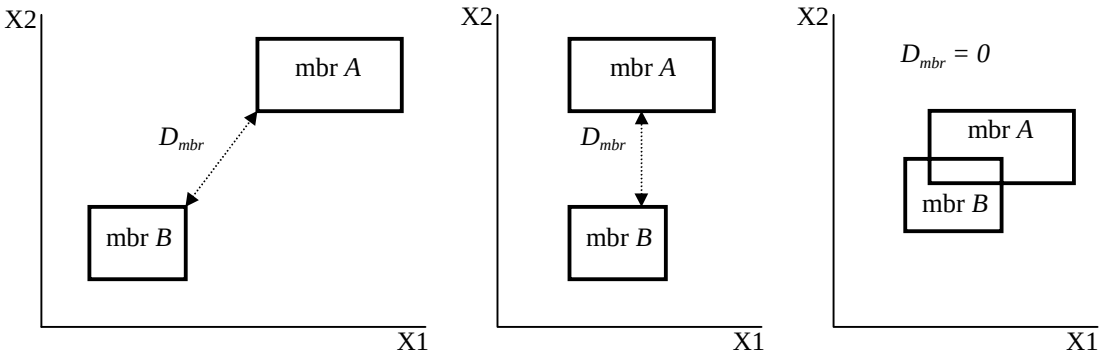

Figure 2. Distance between two MBRs in the 2-dimensional space

$$D(Q,S) = \min_{1\le u\le l-k+1}\left(\sum_{1\le v\le k} d(Q[v], S[u+v-1]) \Big/ k \right) \ge (k*\delta/k) = \delta$$

By Observation 1, $\delta \ge \min_{i\in N,\, j\in R} D_{mbr}(mbr_i(Q), mbr_j(S))$.

Therefore, $D(Q,S) \ge \min_{i\in N,\, j\in R} D_{mbr}(mbr_i(Q), mbr_j(S))$.

So, if $D(Q,S) \le \varepsilon$, there exist a, b such that $a\in N$, $b\in R$, $D_{mbr}(mbr_a(Q), mbr_b(S)) \le \varepsilon$. ■

Based on Lemma 1, we can use the distance D_{mbr} to prune irrelevant sequences from a database without '*false dismissals*,' since it provides the lower bound for the distance between sequences. The next metric we introduce is the normalized distance D_{norm} between two MBRs which considers the number of points in MBR. The distance D_{norm} between a query MBR(mbr_q) and a target MBR(mbr_t) in a data sequence is informally defined as the distance which considers not only the D_{mbr} between mbr_q and mbr_t, but the D_{mbr}'s between mbr_q and the neighboring MBRs of mbr_t when the number of points of mbr_t is less than that of mbr_q. The neighboring MBRs of mbr_t are included one by one in the D_{norm} calculation until the number of points of the participating MBRs reaches that of mbr_q. This distance is intuitively illustrated in Example 2 and the formal definition follows.

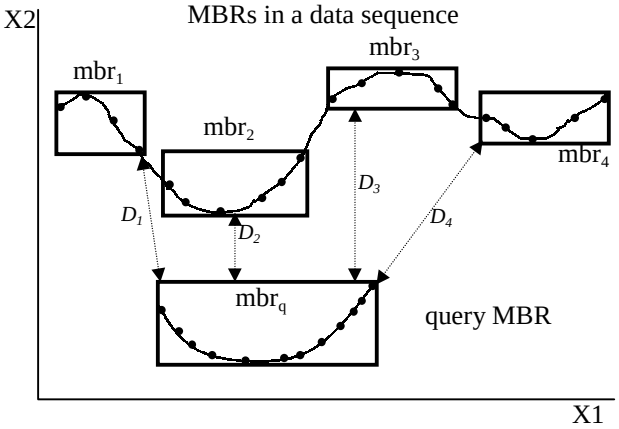

Figure 3. Sketch for the D_{norm} distance

Example 2 Let data sequence S be divided into mbr_j (j=1,2,3,4), D_j ($D_2<D_1<D_3<D_4$) be the MBR distance between mbr_j and the query MBR, the number of points in mbr_j be 4,6,5,5 for j=1,2,3,4 respectively, and the number of points in mbr_q be 12, as shown in Figure 3. Then, the normalized distance $D_{norm}(mbr_q, mbr_2)$ is:

$$D_{norm}(mbr_q, mbr_2) = \frac{\text{weighted distance for } mbr_q}{\text{\# points in } mbr_q} = \frac{(D_2\times 6)+(D_1\times 4)+(D_3\times 2)}{12}$$

If the number of points in mbr_2 is equal or greater than that of mbr_q, the D_2 will be the D_{norm}. Otherwise, one of two adjacent MBRs which has the shorter D_{mbr} (D_1 in this example) is selected. When the total number of points involved in the D_{norm} calculation still does not reach the number of points in mbr_q, then the next adjacent MBR which has the shorter D_{mbr} is selected. This process continues until the condition is satisfied. ■

Definition 5 Let r be the number of MBRs in the data sequence S, and k, l, p, q be the indices of MBRs of S such that $1\le k\le j<l\le r$, $1\le p<j\le q\le r$. Then, the normalized distance D_{norm} between two MBRs is defined as follows:

$$D_{norm}(mbr_i(Q), mbr_j(S)) = \min_{\substack{k\le j<l \\ p<j\le q}} \{LD_{mbr}(mbr_i(Q), \{mbr_k(S),..,mbr_l(S)\}), RD_{mbr}(mbr_i(Q), \{mbr_p(S),..,mbr_q(S)\})\}$$

where, $LD_{mbr}(mbr_i(Q), \{mbr_k(S),.., mbr_l(S)\}) =$

$$\frac{D_{mbr}(mbr_i(Q), mbr_l(S))\cdot(|q_i| - \sum_{t=k}^{l-1}|m_t|) + \sum_{s=k}^{l-1} D_{mbr}(mbr_i(Q), mbr_s(S))\cdot|m_s|}{|q_i|}$$

and, $RD_{mbr}(mbr_i(Q), \{mbr_p(S),.., mbr_q(S)\}) =$

$$\frac{D_{mbr}(mbr_i(Q), mbr_p(S))\cdot(|q_i| - \sum_{t=p+1}^{q}|m_t|) + \sum_{s=p+1}^{q} D_{mbr}(mbr_i(Q), mbr_s(S))\cdot|m_s|}{|q_i|}$$

q_i is the set of points of Q in the i-th MBR, m_j is the set of points of S in the j-th MBR, and

$$\sum_{s=k}^{l-1}|m_s| < |q_i|, \sum_{s=k}^{l}|m_s| \ge |q_i|, \sum_{s=p+1}^{q}|m_s| < |q_i|, \sum_{s=p}^{q}|m_s| \ge |q_i|.$$

The objective of this metric is (1) to provide the greater lower bound than the MBR distance D_{mbr} to promote the pruning of irrelevant sequences, and (2) to make it possible to find the subsequences of each selected sequence, each of which falls under a given threshold. The following two lemmas show that the D_{norm} distance guarantees '*no false dismissal*' to prune irrelevant sequences from a database.

Lemma 2 (Lower Bounding Distance D_{norm} when a query sequence has a single MBR)
Let Q be a query sequence contained in a single MBR $mbr(Q)$ and S' be a subsequence of S, the length of which is the same as that of Q. Since S' is contained in one or more MBRs $mbr_k(S)$ ($l\le k\le s$), then the following holds:

$$\min_{l\le k\le s} D_{norm}(mbr(Q), mbr_k(S)) \le D(Q,S')$$

Proof: Let D_k be an MBR distance between $mbr(Q)$ and $mbr_k(S)$, and m_q, m_k be the set of points in $mbr(Q)$ and $mbr_k(S)$. Then, a set of indices of MBRs in a sequence S involved for the D_{norm} calculation is a subset of $\{l, l+1,..,s\}$. When we let $\hat{m}_k$ be a set of points in $mbr_k(S)$ actually involved for the D_{norm} calculation, then $|m_q| = |\hat{m}_l| + |\hat{m}_{l+1}| + ... + |\hat{m}_s|$. Since each MBR distance D_k is shorter than the distance between any pair of points, one in a sequence Q and the other in a subsequence of S' by Observation 1,

$$\min_{l\le k\le s} D_{norm}(mbr(Q), mbr_k(S)) = \frac{D_l\times|\hat{m}_l| + ... + D_s\times|\hat{m}_s|}{|m_q|} \le D(Q,S')$$

■

Lemma 3 (Lower Bounding Distance D_{norm} when a query sequence has multiple MBRs)
The shortest D_{norm} between any pair of MBRs in a query sequence Q and a data sequence S is the *upper bound* of the shortest D_{mbr} between two sequences, and the *lower bound* of the distance $D(Q,S)$. When N, R is the set of MBRs in Q, S respectively, the following hold:

$$\min_{i\in N, j\in R} D_{mbr}(mbr_i(Q), mbr_j(S)) \le \min_{i\in N, j\in R} D_{norm}(mbr_i(Q), mbr_j(S)) \le D(Q,S)$$

Proof: Let us assume that the D_{norm} has the minimum value at $i=x$ and $j=y$, and a set of indices of MBRs in a sequence S involved for the D_{norm} calculation is $\{l,.., y,.., s\}$. Let q_i, m_j be the set of points in $mbr_i(Q)$ and $mbr_j(S)$. By Definition 5, at the marginal MBR ($j=l$ or $j=s$), the number of points involved in the D_{norm} calculation may be less than $|m_l|$ or $|m_s|$, respectively. Let $|\hat{m}_l|$ and $|\hat{m}_s|$ be the number of points actually involved in the D_{norm} calculation for $mbr_l(S)$ and $mbr_s(S)$. Then,

$|\hat{m}_l| \le |m_l|, |\hat{m}_s| \le |m_s|, \quad |q_x| = |\hat{m}_l| + |m_{l+1}| + ... + |\hat{m}_s|$.

Let

$$D_k = D_{mbr}(mbr_i(Q), mbr_k(S)) \text{ and } \min_{i\in N, j\in R} D_{mbr}(mbr_i(Q), mbr_j(S)) = C_1$$

Then, $\min_{i\in N, j\in R} D_{norm}(mbr_i(Q), mbr_j(S)) = D_{norm}(mbr_x, mbr_y)$

$$= \frac{D_l\times|\hat{m}_l| + ... + D_y\times|m_y| + ... + D_s\times|\hat{m}_s|}{|q_x|} \ge \frac{C_1\times|\hat{m}_l| + ... + C_1\times|m_y| + ... + C_1\times|\hat{m}_s|}{|q_x|} = C_1$$

Therefore,

$$\min_{i\in N, j\in R} D_{mbr}(mbr_i(Q), mbr_j(S)) \le \min_{i\in N, j\in R} D_{norm}(mbr_i(Q), mbr_j(S))$$

Next, we need to show that the shortest D_{norm} between any pair of MBRs in a query and a data sequence is the *lower bound* of the distance $D(Q,S)$. Let the shortest D_{norm} be C_2, and two sequences Q and S have the minimum mean distance at the interval $Q[1{:}k]$ and $S[h{:}h{+}k{-}1]$. Let $Q[1{:}k]$ be partitioned into subsequences Q^i ($1\le i\le n$) which are contained in $mbr_i(Q)$. We also partition $S[h{:}h{+}k{-}1]$ into subsequences S^i ($1\le i\le n$), such that Q^i and S^i are of the same length for $1\le i\le n$. When S^i is contained in $mbr_t(S)$ ($l\le t\le s$), the following holds for a pair of subsequences Q^i and S^i, by Lemma 2.

$$D(Q^i, S^i) \ge \min_{l\le t\le s} D_{norm}(mbr_i(Q), mbr_t(S)) \ge \min_{i\in N, j\in R} D_{norm}(mbr_i(Q), mbr_j(S)) = C_2.$$

Thus, $D(Q,S) = D(Q[1:k], S[h:h+k-1])$

$$= \frac{\sum_{1\le i\le n} |m_i| \times D(Q^i, S^i)}{\sum_{1\le i\le n} |m_i|} \ge \frac{\sum_{1\le i\le n} |m_i| \times C_2}{\sum_{1\le i\le n} |m_i|} = C_2$$

Therefore, we conclude that

$$\min_{i\in N, j\in R} D_{norm}(mbr_i(Q), mbr_j(S)) \le D(Q,S)$$

■

3.3. Finding subsequences

Given a query such as *'Select sequences in a database which contain the subsequences that are similar to a query sequence, and report those subsequences,'* the normalized distance D_{norm} is used to find the answer subsequences, as we mentioned in Section 3.2. Now we need to define the term, *Solution Interval*, the interval of a data sequence which contains subsequences within a given threshold, with respect to a query sequence.

Definition 6 The *Solution Interval* (*SI*) is defined as a set of points which are contained in subsequences, whose distance from a query sequence falls within a specified threshold. That is, the solution interval of $S[1{:}m]$ with respect to a query sequence $Q[1{:}k]$, is formally defined as: (Without loss of generality, we assume $k \le m$.)

$SI = \{S[t] \mid S[t] \in S[p{:}q]$, $S[p{:}q]$ is a subsequence of S such that $D(Q[1{:}k], S[j{:}j{+}k{-}1]) \le \varepsilon$, for $p \le j \le q{-}k{+}1\}$.

The solution interval in Definition 6 describes the actual answer subsequences that can be obtained by a sequential scan. To avoid an overhead of the sequential scan, we will approximate it by using the normalized distance D_{norm}. When we calculate the D_{norm} between $mbr_q(Q)$ and $mbr_s(S)$, one or more MBRs of a data sequence are involved. If $D_{norm}(mbr_q(Q), mbr_s(S)) \le \varepsilon$, we can approximate the solution interval as the set of all points which are involved in the D_{norm} calculation. The example 3 illustrates it intuitively.

Example 3 Figure 3 in Section 3.2 shows that the mbr_1, mbr_2, and mbr_3 of a data sequence are involved to calculate the normalized distance $D_{norm}(mbr_q, mbr_2)$. When $D_{norm}(mbr_q, mbr_2) \le \varepsilon$, we can approximate the solution interval *SI* as a subsequence of a data sequence which is composed of 4 points of mbr_1, 6 points of mbr_2, and *first* 2 points of mbr_3. That is, *SI* = {*all points* contained in mbr_1, mbr_2} ∪ {*first* 2 points of mbr_3}. ■

While the D_{norm} metrics guarantees '*no false dismissal*' for selecting candidate sequences from a database, it does not guarantee '*no false dismissal*' for determining the solution interval to find subsequences of each selected sequence. However, the experimental result in Section 4 shows that the determination demonstrates '*almost no false dismissal*' with the recall over 98%. The false dismissal arises from the fact that few points between subsequences involved in the D_{norm} calculation may be missed. Therefore, a small amount of false dismissal does not cause a severe problem. In a real situation, we can choose the interval for playing a selected video based on an approximated solution interval, by containing few false-dismissed points. Now, we are ready to describe the proposed algorithm for the similarity search of multidimensional data sequences.

3.4. Similarity search processing

In this section, we present the overall process of our proposed method. Before a query is processed, we shall do some pre-processing to extract the feature vectors from raw materials such as video streams or images, and then to construct an index for later processing. After the index is constructed, the similarity search is performed to select candidate sequences from a database (*Phase* 2 and 3), and to find the solution interval of each selected sequence (*Phase* 3).

3.4.1. Pre-processing

1. *Generation of multidimensional sequences.* Raw materials are parsed to extract the feature vectors. Each vector is represented by a multidimensional point in the hyper data space. When the vector is of high dimension, various dimension reduction techniques such as DFT or Wavelets can be applied to avoid the dimensionality curse problem. A series of vectors constitutes a multidimensional sequence.
2. *Index construction.* Each multidimensional sequence is partitioned into subsequences by using the partitioning algorithm in Section 3.4.3. Each subsequence is enclosed by an MBR. Every MBR is indexed and stored into a database by using any R-tree variant.

3.4.2. Similarity search

After an index has been constructed, we perform the similarity search against a given query sequence. A three-phase algorithm for the similarity search is described below.

Algorithm SIMILARITY_SEARCH

/ i is an index for MBRs in Q, j is an index for MBRs in a data sequence S, and k is an index for a data sequence in a database */*
AS_{mbr} := {} */* the set of answer sequences by D_{mbr} */*
AS_{norm} := {} */* the set of answer sequences by D_{norm} */*
SI_k := {} */* the solution interval of the sequence S_k */*
/ By using the partitioning algorithm in Section* 3.4.3. **/*
Phase 1. Partitioning of a query sequence
Partition query sequence Q into one or more $mbr_i(Q)$
Phase 2. First pruning. (Index search)
For each $mbr_i(Q)$ of query sequence Q,
  Search a database
    If $D_{mbr}(mbr_i(Q), mbr_j(S_k)) \le \varepsilon$ Then
      Insert S_k into the set AS_{mbr}
Phase 3. Second pruning and solution interval finding
For each selected sequence S_k in the set AS_{mbr}
  For each $mbr_i(Q)$ of a query sequence
    If $D_{norm}(mbr_i(Q), mbr_j(S_k)) \le \varepsilon$ Then
      Insert S_k into the set AS_{norm}
      SI_k = {all points which are involved in the D_{norm} calculation} $\cup$ SI_k
      Report the set AS_{norm} and the set SI_k

3.4.3. Partitioning of a sequence

A multidimensional sequence is partitioned into subsequences, each of which is contained in an MBR. To do this, our method uses the partitioning algorithm proposed in [5] with a slight modification of its cost function. The algorithm uses a cost function, which estimates the number of disk accesses for an MBR. They defined the marginal cost (*MCOST*) of a point in an MBR as the average number of disk accesses (DA) divided by the number of points in the MBR. The grouping of points into subsequences is done in such a way that if the *MCOST* is increased for a successive point of a sequence, then another subsequence is started from the point, otherwise it is included in the current subsequence. We slightly modified *MCOST* to reflect the MBRs of a query sequence in addition to the MBRs of a data sequence, since the similarity search is based upon the MBRs of both a query sequence and a data sequence in our proposed method. Consider an n-dimensional subsequence with m points, an enclosing MBR of which has the sides $L=(L_1,L_2,\ldots,L_n)$. If we assume that a query MBR has the sides $Q=(Q_1,Q_2,\ldots,Q_n)$ and a given threshold is ε, then the *MCOST* of each point in this MBR will be:

$$MCOST = \frac{DA}{m} = \frac{\prod_{k=1}^{n}(L_k + Q_k + \varepsilon)}{m}$$

By evaluating various combinations of Q_k and ε, we can choose an appropriate value for $Q_k+\varepsilon$. We adopt 0.3 for this value, since it demonstrates the best partitioning by an extensive experiment. The partitioning algorithm is described as follows:

Algorithm PARTITIONING_SEQUENCE

P := {} */* the set of points in sequence S */*
M := {} */* the set of MBRs in sequence S */*
count : *the number of points in MBR*
max : *the predefined value of maximum points per MBR*
Allocate the first point of P in current MBR
For each point in P
  If it increase *MCOST* or *count* > *max* Then
    Insert current MBR into M
    Start another MBR
  else
    Insert it into current MBR
    Increment *count*

4. Experiments

In order to measure the effectiveness and performance of our proposed method, we have conducted comprehensive experiments on real video data sets, as well as synthetic data sets generated by using a Fractal function. The video data sets include a collection of TV news, dramas, and documentary films. Our experiment focuses on showing the efficiency of the method to prune a

database of irrelevant sequences and to find the solution interval. The system is written in Microsoft VC++ under Windows NT, on a HP NetServer.

4.1. Generation of experimental data

For the experiment, we have used synthetic multi-dimensional data sequences and real video data stream. All data sets are, for convenience, 3-dimensional, but any dimensional data sequences can be used. Synthetic data sequences are obtained using a Fractal function as follows:

1. Two 3-dimensional initial points, P_{start} for the beginning point of a sequence and P_{end} for the end point, are selected randomly in the 3-dimensional unit cube $[0,1)^3$.
2. The middle point P_{mid} of P_{start} and P_{end} is calculated as follows:
   $$P_{mid} = (P_{start} + P_{end}) / 2 + dev*random()$$
   where the *dev* is selected to control the amplitude of a sequence in the range [0,1), and *random*() is the function for generating a random real number from 0 to 1.
3. After getting P_{mid}, two subsequences (P_{start}, P_{mid}) and (P_{mid}, P_{end}) are generated. The process 2 is repeated recursively for each subsequence, with a new *dev*.
   $$dev = scale * dev$$
   where *scale* is a real number in the range [0,1). We use a lower *dev* value for the successive recursive call, since the lengths of the two subsequences are shorter than their parent.

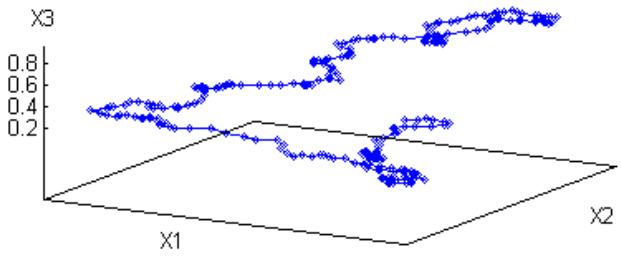

Figure 4. A synthetically generated sequence

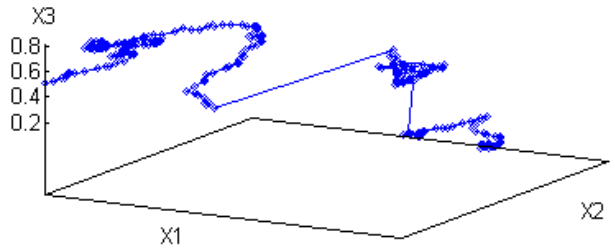

Figure 5. A sequence generated from video data

Video data streams are generated from various video data sources, by extracting color features from each frame. Thus, each frame is mapped to a 3-dimensional point in the unit cube. Figure 4 and 5 show the sample sequences generated synthetically and from a real video stream.

The following table summarizes the parameter setting used in the experiment.

Table 2. Experimental parameters

	Synthetic data	Real video data
# of data sequences	1600	1408
Length of data sequences	arbitrary (56-512 points)	arbitrary (56-512 frames)
Range of threshold values(ε)	0.05-0.50	0.05-0.50
# of query sequences for each ε	20	20

4.2. Experimental results

The range search is to find similar sequences in a database to a given query sequence within a specified threshold. We have executed test cases which are composed of various combinations of test parameters. The threshold range 0.05-0.50 is chosen since it provides enough coverage for the low and high selectivity in the $[0,1)^3$ cube which has a diagonal of $\sqrt{3}$. A query diameter will be 0.1 through 1.0. For each test, we have issued randomly selected 20 queries and taken the average of query results. Several aspects of our method are examined in the experiment.

4.2.1. Pruning efficiency for selecting sequences in a database

The pruning efficiency of the D_{mbr} and D_{norm} to select candidate sequences from a database was examined by varying the threshold value. To measure it, we define the *pruning rate* (*PR*) as follows:

$$\mathrm{PR} = \frac{|\text{sequences actually pruned}|}{|\text{sequences to be pruned}|} = \frac{|\text{total seq.}| - |\text{retrieved seq.}|}{|\text{total seq.}| - |\text{relevant seq.}|}$$

where $|S|$ denotes the number of elements in the set S.

Figure 6 and 7 show the results by using the metrics defined in this paper for synthetic and real video data sets. We can observe that the pruning rate of the D_{mbr} is 70-90% on synthetic data sets and 65-91% on video data sets for the range of a threshold value, 0.05 through 0.50. The pruning rate of the D_{norm} shows constantly 3-10 % better than that of the D_{mbr} in the whole range, 76-93% for synthetic data sets and 73-94% for the video data sets. The experiment also shows that the pruning rate decreases as a given threshold value increases. This is because the number of retrieved sequences for a larger threshold increases more rapidly than the number of relevant sequences.

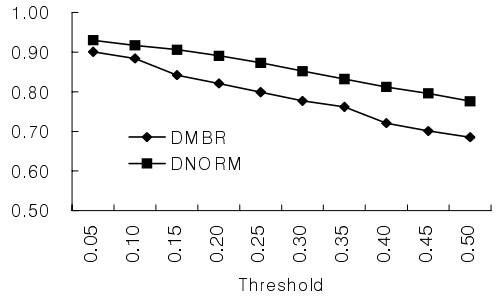

Figure 6. Pruning rate of the D_{mbr} and the D_{norm} for synthetic data sets

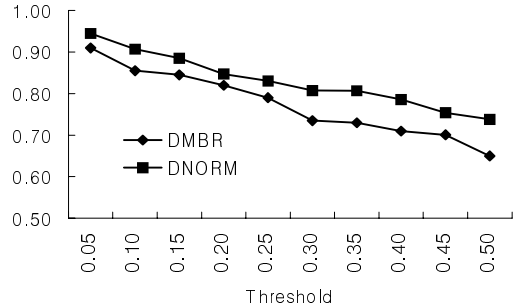

Figure 7. Pruning rate of the D_{mbr} and the D_{norm} for real video data sets

4.2.2. Pruning efficiency for finding subsequences

We have examined the pruning efficiency of the solution interval estimated by our proposed method. First, we present the measures to estimate the efficiency of the selected solution interval. Let P_{total} be a set of total points in a sequence, P_{scan} be the answer set of points by using the sequential scan, and P_{norm} be the candidate set of points by using the D_{norm} metrics. Then, the pruning efficiency of the solution interval PR_{SI} is defined as follows:

$$PR_{SI} = \frac{|\text{points actually pruned}|}{|\text{points to be pruned}|} = \frac{|P_{total}| - |P_{norm}|}{|P_{total}| - |P_{scan}|}$$

As the *solution interval* computed by using the D_{norm} metrics does not guarantee the correctness, we need to examine the *Recall* of a selected solution interval. It is defined as follows:

$$\text{Recall} = \frac{|P_{scan} \cap P_{norm}|}{|P_{scan}|}$$

Figure 8 and 9 show the results on the pruning efficiency and the recall of the estimated solution interval. First, we can observe that the recall values for both synthetic and real video data sets are very close to 1.0, in the range of 98-100%. It indicates that our method has almost 'no false dismissal.' The pruning rate PR_{SI} is around 60-80 % on synthetic data sets and 67-94 % on video data sets for the thresholds with the range 0.05 through 0.50. As we can see, the pruning rate of video data sets is better than that of synthetic data sets. It seems that video streams are well clustered than synthetic data sets as shown in Figure 4 and 5. It is known that the frames in the same shot of a video stream have very similar feature values. Consequently, our method demonstrates almost 1.0 recall values, and 60-94% pruning rate for given thresholds.

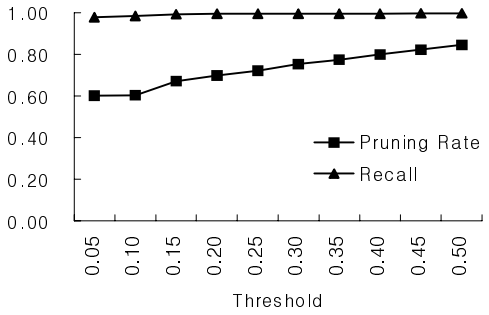

Figure 8. Efficiency of the solution interval for synthetic data sets

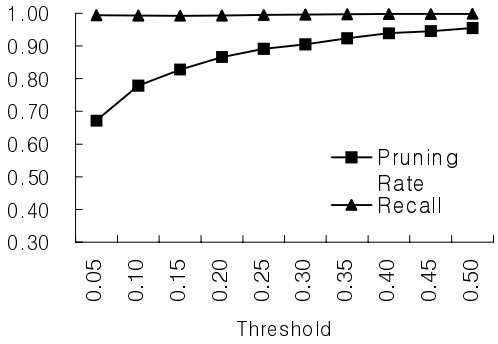

Figure 9. Efficiency of the solution interval for real video data sets

4.2.3. Average Response Time

We have compared the response time of our method with the sequential scan. We have run 20 random queries with the same threshold over synthetic and real data sets to find similar sequences and solution intervals with given sequences, by using the D_{mbr} and D_{norm} metrics. The results were compared against the sequential scan. The ratio of the response time is obtained as follows:

$$\text{Response time ratio} = \frac{\text{Response time by the sequential scan}}{\text{Response time by the proposed algorithm}}$$

where a denominator represents the total elapsed time for the search by both the D_{mbr} and D_{norm} metrics, since we consider the total time to select candidate data sequences and to get the estimated solution interval.

Figure 10 depicts the ratio of the average response time of our method compared to the sequential scan. We can observe that our method performs 22-28 times better for synthetic data sets and 16-23 times better for real video data sets than the sequential scan.

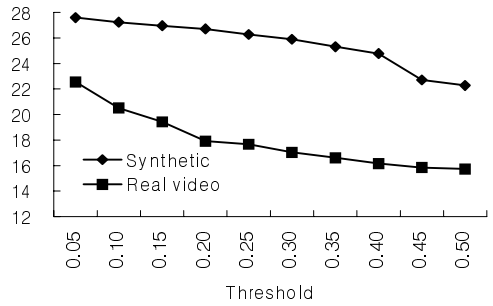

Figure 10. Response time ratio for synthetic and real video data sets

5. Conclusions

In this paper, we have focused on retrieving similar multidimensional sequences, such as video streams, from a large database. To solve the problem, we have first defined the distance between two multidimensional sequences, and introduced two lower bounding distance metrics, D_{mbr} and D_{norm}. Based on the metrics, we proposed the algorithm to prune a database of irrelevant sequences and to find the solution interval of the selected sequences. One of potential applications which is emphasized in this paper is the similarity query on large video data sets, but we believe other application areas can also benefit. The desirable characteristics of our method are summarized as follows:

1. The search algorithm is based upon MBRs, not each point in sequences. Thus, it is fast and needs small storage overhead compared with a sequential scan.
2. Our method is designed to handle the sophisticated similarity search, such as finding subsequences of a selected sequence, as well as selecting just candidate sequences from a database.
3. Our method does not restrict the length of sequences. Query sequences and data sequences are of arbitrary length. In particular, a given query sequence may be longer than a data sequence.
4. Any multidimensional access method (such as the R*-tree or the X-tree) can be used.

Our major contribution is that the traditional similarity search method on one-dimensional time-series data is extended to support multidimensional data sequences, a more generalized format of sequences. We have performed an experiment with synthetic and real video data sets, and examined the pruning efficiency and performance of our proposed method over the sequential scan. Our method has shown a remarkable efficiency to prune irrelevant sequences and to find the solution interval of selected sequences, consequently the response time is 16-28 times faster than that of the sequential scan.

References

[1] R. Agrawal, C. Faloutsos, A. Swami. Efficient Similarity Search in Sequence Databases. *Proceedings of Foundations of Data Organizations and algorithms* (*FODO*), pages 69-84, Evanstone, Illinois, October 1993.

[2] S.Berchtold, C. Bohm, H.Kriegel. The Pyramid-Technique: Towards Breaking the Curse of Dimensionality. *Proceedings of ACM SIGMOD Int'l Conference on Management of Data*, pages 142-153, Seattle Washington, June 1998.

[3] S. Berchtold, D. Keim, H. Kriegel. The X-tree: An Index Structure for High-Dimensional Data. *Proceedings of Int'l Conference on Very Large Data Bases*, pages 28-39, Bombay, India, September 1996.

[4] N. Beckmann, H. Kriegel, R. Schneider, B. Seeger. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. *Proceedings of ACM SIGMOD Int'l Conference on Management of Data*, pages 322-331, Atlantic City, New Jersey, May 1990.

[5] C. Faloutsos, M. Ranganathan, Y. Manolopoulos. Fast Subsequence Matching in Time-Series Databases. *Proceedings of ACM SIGMOD Int'l Conference on Management of Data*, pages 419-429, Minneapolis, Minnesota, May 1994.

[6] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, P. Yanker. Query by Image and Video Content: The QBIC System. *IEEE Computer*, Vol. 28, No. 9, pages 23-32, September 1995.

[7] A.Guttman. R-trees: a dynamic index structure for spatial searching. *Proceedings of ACM SIGMOD Int'l Conference on Management of Data*, pages 47-57, Boston, Massachusetts, June, 1984.

[8] H.V. Jagadish. Linear Clustering of Objects with Multiple Attributes. *Proceedings of ACM SIGMOD Int'l Conference on Management of Data*, pages 332-342, Atlantic City, New Jersey, May 1990.

[9] N.Katayama, S. Satoh. The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries. *Proceedings of ACM SIGMOD Int'l Conference on Management of Data*, pages 369-380, Tucson, Arizona, May 1997.

[10] D. Rafiei, On Similarity Queries for Time Series Data. *Proceedings of Int'l Conference on Data Engineering*, pages 410-417, Sydney, Australia, March 1999.

[11] N. Roussopoulos, S. Kelley, F. Vincent. Nearest Neighbor Queries. *Proceedings of ACM SIGMOD Int'l Conference on Management of Data*, pages 71-79, San Jose, California, May 1995.

[12] D. Rafiei, A. Mendelzon. Similarity-Based Queries for Time Series Data. *Proceedings of ACM SIGMOD Int'l Conference on Management of Data*, pages 13-25, Tucson, Arizona, May 1997.

[13] B. Yi, H.Jagadish, C. Faloutsos. Efficient Retrieval of Similar Time Sequences Under Time Warping. *Proceedings of Int'l Conference on Data Engineering*, pages 201-208, Orlando, Florida, February 1998.