# Symbolic Representation and Retrieval of Moving Object Trajectories

Lei Chen, M. Tamer Özsu
University of Waterloo
School of Computer Science
Waterloo, Canada
{l6chen,tozsu}@uwaterloo.ca

Vincent Oria
New Jersey Inst. of Technology
Dept. of Computer Science
Newark, New Jersey, USA
vincent.oria@njit.edu

## ABSTRACT

Searching moving object trajectories of video databases has been applied to many fields, such as video data analysis, content-based video retrieval, video scene classification. In this paper, we propose a novel representation of trajectories, called *movement pattern strings*, which convert the trajectories into symbolic representations. Movement pattern strings encode both the movement direction and the movement distance information of the trajectories. The distances that are computed in a symbolic space are lower bounds of the distances of original trajectory data, which guarantees that no false dismissals will be introduced using movement pattern strings to retrieve trajectories. In order to improve the retrieval efficiency, we define a *modified frequency distance* for frequency vectors that are obtained from movement pattern strings to reduce the dimensionality and the computation cost. The experimental results show that using movement pattern strings is almost as effective as using raw trajectories. In addition, the cost of retrieving similar trajectories can greatly be reduced when the modified frequency distance is used as a filter.

**Categories and Subject Descriptors:** H.2.4 [DATABASE MANAGEMENT]: Systems - multimedia database; H.3.3 [INFORMATION STORAGE AND RETRIEVAL]: Information Search and Retrieval - information filtering

**General Terms:** Algorithms, Experimentation

**Keywords:** Trajectory, Symbolic representation, Movement pattern string, Edit distance on real sequences

## 1. INTRODUCTION

Recently, a number of interesting applications have been developed based on the analysis of trajectories of moving objects in videos. For example, In sports videos, such as hockey, it is quite useful for coaches or sports researchers to know the movement patterns of top players. In a store surveillance video monitoring system, finding the customers' movement patterns may help in the arrangement of merchandise. All of these applications require the definition of accurate and robust similarity measures to determine the similarity among trajectories. A number of distance functions have been proposed, such as Euclidean distance [2, 14], Dynamic Time Warping (DTW) [5], Edit distance with Real Penalty (ERP) [3], but they are sensitive to noise, shifts and scaling of data that commonly occur due to capturing device failures, errors in detection techniques, and different sampling rates. Even though Longest Common Subsequence (LCSS) [19] is robust to noise, it is not always accurate. We proposed a novel distance function, Edit Distance on Real Sequence (EDR) to handel these problems [4]. EDR is based on the Edit Distance (ED) on strings [15] which is widely used in bio-informatics and speech recognition to measure the similarity between two strings. Analysis and experimental comparison of EDR with Euclidean distance, DTW, ERP, and LCSS, indicate that EDR is more robust than Euclidean distance, DTW and ERP, and it is more accurate than LCSS.

However, the computation cost of EDR on trajectories is quadratic; this time cost becomes critical when the trajectory database is large. Since ED was originally defined on strings for which many algorithms, data structures, and embedded edit distance functions have been developed, it is intuitive to think about the possibility of converting real valued movement sequences into symbolic representation and applying the string dimensionality reduction techniques to reduce the computation and retrieval cost. Therefore, in this paper, we propose a symbolic representation of trajectories, movement pattern string (MPS). Furthermore, we define a distance function, *Edit Distance on MPS* (EDM), on the converted symbolic space and prove that EDM is a lower bound of EDR on original space. This lower bounding property guarantees that conducting a similarity search on MPS will not introduce false dismissals [6]. Finally, we define a *modified frequency distance* (MFD) between two *frequency vectors* (FV) of MPSs to reduce the cost of CPU time on computing EDR of two trajectories.

The main contributions of our paper are the following:

1. We develop a transformation scheme to convert a trajectory into a symbolic representation, Movement Pattern String (MPS). Compared to raw representation of trajectories, MPSs need less storage space, and most importantly, our experimental results show that MPSs are almost as effective as raw representation in classification and clustering tasks.

*MIR'04,* October 15–16, 2004, New York, New York, USA.
Copyright 2004 ACM 1-58113-940-3/04/0010 ...$5.00.

2. We prove that EDM, computed over a symbolic space, is a lower bound of EDR on real trajectory data.

3. We propose a modified frequency distance, MFD, as a distance measure of frequency vectors that are obtained from the MPSs. We also prove that MFD on frequency vectors is a lower bound of EDM on the corresponding MPSs.

The rest of the paper is arranged as follows: Section 2 introduces the distance measure EDR, and also briefly presents some definitions. In Section 3, we present our symbolic representation of trajectories and prove the lower bound property. Section 4 introduces the modified frequency distances followed by the retrieval efficiency study of our symbolic representation and frequency vectors in Section 5. Section 6 provides a in depth comparison with the related work. We conclude in Section 7 and indicate some further work.

## 2. EDIT DISTANCE ON REAL SEQUENCE

In this section, we first give the formal definitions of trajectories, sequences of (movement direction, distance ratio) pairs, we then briefly introduce Edit Distance on Real sequence between two trajectories proposed in [4].

### 2.1 Preliminaries

In the following, we assume that objects are points that move in a two-dimensional space ($x-y$ plane) and that time is discrete.

**Definition 1.** Given a moving object $A$, its *trajectory*, $T_A$, is defined as a sequence of coordinates that are pairs showing the position of an object $A$ in the $x-y$ plane: $T_A = [(x_{a,1}, y_{a,1}), \ldots, (x_{a,n}, y_{a,n})]$, where $n$, the number of positions in $T_A$, is defined as the *length* of $T_A$.

We refer to $T_A$ as the *raw representation of the trajectory*, since this is the most likely data format that we can get from tracing sensor or extraction techniques. However, directly using this raw representation to compare trajectories will miss the trajectories with similar movement but different spatial rotation, shifting, or scaling factors. For example, in Figure 1, three trajectories are shown: $T_B$ can be derived from $T_A$ by scaling its $x$ and $y$ positions by a factor of 2, while $T_C$ is translated from $T_B$ by shifting its $x$ and $y$ positions by 1. The three trajectories have similar movement patterns, however, comparing their raw representations, (which are: $T_A = [(3.5, 4.5), (1.5, 2.5), (2.5, 3.5), (2, 3), (3, 4)]$, $T_B = [(7, 9), (3, 5), (5, 7), (4, 6), (6, 8)]$, and $T_C = [(8, 10), (4, 6), (6, 8), (5, 7), (7, 9)]$.) can not lead to a conclusion that they are similar unless scaling or shifting factors are introduced in the similarity measures. However, this will increase the cost for computing the similarity measure (e.g. in [19], by only introducing the shifting factors in LCSS, the computation cost is increased by $O(n)$, where $n$ is the length of the trajectory).

Thus, instead of directly using raw representation of trajectories, we represent them by means of a sequence of (movement direction, distance ratio) pairs. This representation is not affected by rotation, shifting or scaling [18].

**Definition 2.** Given a moving object $A$ and its trajectory $T_A$ of length $n$ ($n > 1$), the *sequence of (movement direction, distance ratio) pair* $M_A$ is defined as a sequence of pairs: $M_A = [(\theta_{a,1}, \sigma_{a,1}), \ldots, (\theta_{a,m}, \sigma_{a,m})]$, where $m = n - 1$ and $n$ is the length of $T_A$. The *movement direction* $\theta_{a,i}$ is

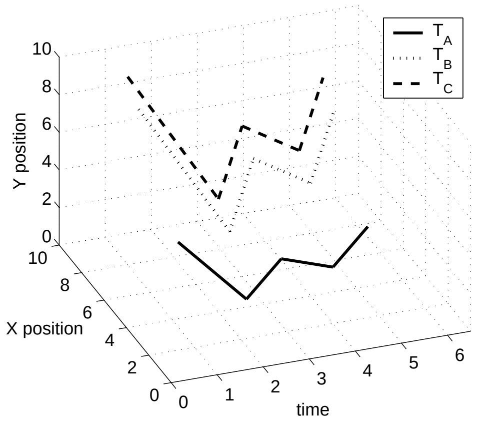


**Figure 1: The trajectories with different scaling and shifting factors share similar movement pattern**

$$\theta_{a,i} = \begin{cases} \arctan(\frac{y_{a,(i+1)} - y_{a,i}}{x_{a,(i+1)} - x_{a,i}}) & x_{a,(i+1)} - x_{a,i} \geq 0, \\ \arctan(\frac{y_{a,(i+1)} - y_{a,i}}{x_{a,(i+1)} - x_{a,i}}) - \pi & y_{a,(i+1)} - y_{a,i} \leq 0 \text{ and} \\ & x_{a,(i+1)} - x_{a,i} < 0, \\ \arctan(\frac{y_{a,(i+1)} - y_{a,i}}{x_{a,(i+1)} - x_{a,i}}) + \pi & y_{a,(i+1)} - y_{a,i} > 0 \text{ and} \\ & x_{a,(i+1)} - x_{a,i} < 0. \end{cases}$$

The *movement distance ratio* $\sigma_{a,i}$ is

$$\sigma_{a,i} = \begin{cases} \frac{\sqrt{(y_{a,(i+1)} - y_{a,i})^2 + (x_{a,(i+1)} - x_{a,i})^2}}{TD(T_A)} & TD(T_A) \neq 0 \\ 0 & 0 \end{cases}$$

where the <u>total movement distance</u> of $T_A$ is $TD(T_A) = \sum_{1 \leq j \leq n-1} \sqrt{(y_{a,(j+1)} - y_{a,j})^2 + (x_{a,(j+1)} - x_{a,j})^2}$.

Base on this definition, we know that $\theta_{a,i}$ ranges from $-\pi$ to $\pi$ and $\sigma_{a,i}$ ranges from 0 to 1. We use $M_A(n)$ to denote the sequence $[(\theta_{a,1}, \sigma_{a,1}), \ldots, (\theta_{a,n}, \sigma_{a,n})]$ where $n$ is the length of the sequence. All three examples of trajectories in Figure 1 have the same sequence of (movement direction, distance ration) pairs, which conforms to the fact that they have similar movement patterns.

In the rest of this paper, we use the terms "movement sequence" or "sequence" interchangeably to refer to the sequence of (movement direction, distance ratio) pairs unless specified otherwise.

### 2.2 Edit Distance on Real sequence

*Edit Distance on Real Sequence* (EDR) between two movement sequences counts similar subsequences and assigns penalties to the gaps in between these subsequences, thus, unlike LCSS, it considers gaps within sequences [4]. EDR is based on ED ([15] on strings. Given two strings $A$ and $B$, $ED(A, B)$ is the number of insert, delete, or replace operations that are needed to change $A$ into $B$. Since movement sequences are not strings but numerical value pair sequences, we have to define the cases which element pairs of different movement sequences *match*. We introduce two thresholds $\epsilon_{dir}$ and $\epsilon_{dis}$ to that end: $\epsilon_{dir}$ is used to determine whether two movement directions match and $\epsilon_{dis}$ is for determining the similarity of two movement distance ratios.

**Definition 3.** Two (movement direction, distance ratio) pairs $(\theta_{a,i}, \sigma_{a,i})$ and $(\theta_{b,j}, \sigma_{b,j})$ are said to *match* if and only if $|\theta_{a,i} - \theta_{b,j}| \leq \epsilon_{dir}$ and $|\sigma_{a,i} - \sigma_{b,j}| \leq \epsilon_{dis}$. This is specified as predicate $match((\theta_{a,i}, \sigma_{a,i}), (\theta_{b,j}, \sigma_{b,j}))$.

Based on the definition of *match*, the EDR between two movement sequences is defined as follows:

**Definition 4.** Given two moving objects $A$ and $B$ and their movement sequences $M_A$ of length $n$ and $M_B$ of length $m$, respectively, the EDR between $M_A$ and $M_B$ is the number of insert, delete, or replace operations that is needed to change $M_A$ into $M_B$. $EDR(M_A(n), M_B(m))$ can be computed as follows:

$$\begin{cases} n & \text{if } m = 0; \\ m & \text{if } n = 0; \\ EDR(M_A(n-1), M_B(m-1)) & \\ \quad\quad \text{if } match((\theta_{a,i}, \sigma_{a,i}), (\theta_{b,j}, \sigma_{b,j})) & \\ min[EDR(M_A(n-1), M_B(m-1)) + 1, & \\ \quad EDR(M_A(n-1), M_B(m)) + 1, & \\ \quad EDR(M_A(n), M_B(m-1)) + 1] & \\ \quad\quad \text{otherwise} & \end{cases}$$

Although EDR is more accurate, its computation still costs $O(n \times m)$ time and space with dynamic programming, where $n$ and $m$ are the lengths of the two sequences $M_A$ and $M_B$, respectively. In next two sections, we will propose a symbolic representation of trajectories, discuss the computation cost of the symbolic trajectories and propose an improvement of the retrieval efficiency without affecting the accuracy.

## 3. SYMBOLIC REPRESENTATION OF TRAJECTORIES

In this section, we propose a symbolic representation of movement sequences. The basic idea is to quantize the (movement direction, distance ratio) space and represent each subregion by a distinct symbol.

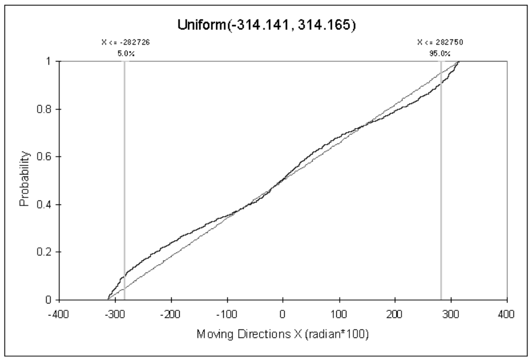
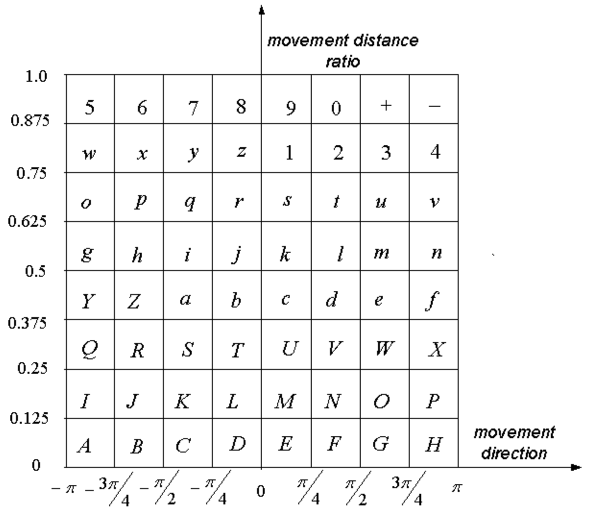


**Figure 2: Probability distribution of movement directions of hokey players**

As defined in Section 2, the values of (movement direction, distance ratio) of an object range from $-\pi$ to $\pi$ and 0 to 1, respectively. We investigated the movement direction distribution of three trajectory data sets, which are: hockey players' trajectories that are extracted from National Hockey League (NHL) videos, the "Cameramouse", and Australian Sign Language data sets (the last two data sets are explained in detail in Section 5). We found that movement directions of all three data sets are uniformly distributed. Figure 2 shows the values of the probability density function of movement directions of NHL data, which shows that the movement direction of a hockey player at each sampled position follows an uniform distribution. Based on these observations, we divide the (movement direction, distance ratio) space into equal sized subregions.



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1.0–0.875 | 5 | 6 | 7 | 8 | 9 | 0 | + | − |
| 0.875–0.75 | w | x | y | z | 1 | 2 | 3 | 4 |
| 0.75–0.625 | o | p | q | r | s | t | u | v |
| 0.625–0.5 | g | h | i | j | k | l | m | n |
| 0.5–0.375 | Y | Z | a | b | c | d | e | f |
| 0.375–0.25 | Q | R | S | T | U | V | W | X |
| 0.25–0.125 | I | J | K | L | M | N | O | P |
| 0.125–0 | A | B | C | D | E | F | G | H |
| | $-\pi$ – $-3\pi/4$ | $-3\pi/4$ – $-\pi/2$ | $-\pi/2$ – $-\pi/4$ | $-\pi/4$ – 0 | 0 – $\pi/4$ | $\pi/4$ – $\pi/2$ | $\pi/2$ – $3\pi/4$ | $3\pi/4$ – $\pi$ |

**Figure 3: An example of (movement direction, distance ratio) quantization map**

Given $\epsilon_{dir}$ and $\epsilon_{dis}$[1], we equally divide the two dimensional (movement direction, distance ratio) space into $(2\pi/\epsilon_{dir}) \times (1.0/\epsilon_{dis})$ subregions and assign each subregion a distinct symbol. The whole set of symbols makes up the *movement pattern alphabet*, which we denote as $\mathcal{A} = A_1, A_2, \ldots, A_s$, where the size of the movement pattern alphabet is $s = (2\pi/\epsilon_{dir}) \times (1.0/\epsilon_{dis})$. Once the two threshold values $\epsilon_{dir}$ and $\epsilon_{dis}$ are given for the trajectory data, the size of movement pattern alphabet is fixed. $A_i$ is a distinct symbol that represents a subregion $SB_i$ of size $\epsilon_{dir} \times \epsilon_{dis}$ $(1 \leq i \leq s)$. Each subregion $SB_i$ is represented by two (movement direction, distance ratio) pairs: $(\theta_{bl,i}, \sigma_{bl,i})$ and $(\theta_{ur,i}, \sigma_{ur,i})$, which are the bottom left and upper right coordinates of $SB_i$.

A *quantization map* (QM) that contains all the subregions and associated symbols is stored as a lookup table, and is used to convert (movement direction, distance ratio) pairs into symbols and to obtain the neighbors for a given symbol. The second function of QM is used to compute the distance in the converted symbolic space (Definitions 6 and 7). Figure 3 gives an example quantization map, which divides (movement direction, distance ratio) space into 64 subregions, each subregion corresponding to a movement symbol[2].

Once we quantize the (movement direction, distance ratio) space into subregions and derive the movement alphabet $\mathcal{A}$, we map a (movement direction, distance ratio) pair $(\theta, \sigma)$ into a symbol. For example, according to the quantization map in Figure 3, $(\frac{\pi}{3}, 0.16)$, $(\frac{\pi}{4}, 0.1)$, and $(-\frac{\pi}{4}, 1.0)$ are mapped into symbols 'N', 'E', and '7', respectively.

[1]The values of $\epsilon_{dir}$ and $\epsilon_{dis}$ are application dependent. They must be given beforehand to determine weather two elements match when we use EDR as a similarity measure to query the trajectory data set.

[2]In Figure 3,$\epsilon_{dir} = \pi/4$ and $\epsilon_{dis} = 0.125$.

**Definition 5.** Given a movement sequence $M_A = [(\theta_{a,1}, \sigma_{a,1}), \ldots, (\theta_{a,n}, \sigma_{a,n})]$ of length $n$ and a movement pattern alphabet $\mathcal{A}$, a *movement pattern string* (MPS) is defined as a sequence of symbols: $S_{a,1}S_{a,2}\ldots S_{a,n}$, where each symbol $S_{a,i}$ $(1 \leq i \leq n)$ is mapped from the movement direction and distance pair $(\theta_{a,i}, \sigma_{a,i})$ according to $\mathcal{A}$.

MPS retains the order of movement sequences by arranging the corresponding symbols from left to right. We use $MPS_A(n)$ to denote the string $S_{a,1}S_{a,2}\ldots S_{a,n}$. Figure 4 gives an example of converting a movement sequence $M_A = [(\frac{\pi}{3}, 0, 16), (-\frac{\pi}{4}, 0, 16), (\frac{\pi}{6}, 0.33), (-\frac{\pi}{3}, 0.33)]$ of $T_A$ to the MPS "NKUS" using movement pattern alphabet as given in Figure 3. It is obvious that the MPS representation is more compact.

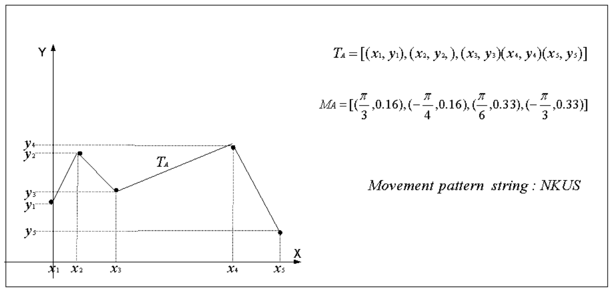

Y
$T_A = [(x_1, y_1), (x_2, y_2,), (x_3, y_3)(x_4, y_4)(x_5, y_5)]$
$M_A = [(\frac{\pi}{3}, 0.16), (-\frac{\pi}{4}, 0.16), (\frac{\pi}{6}, 0.33), (-\frac{\pi}{3}, 0.33)]$
$y_4$
$y_2$
$T_A$
Movement pattern string : NKUS
$y_3$
$y_1$
$y_5$
X
$x_1$
$x_2$
$x_3$
$x_4$
$x_5$

**Figure 4: An example of converting a trajectory to a MPS**

After converting the movement sequences to MPSs, we can think of directly applying ED on two MPSs since they are strings. However, using the standard ED [15] will not provide correct answers, since the ED computed on the MPSs is not a lower bound of EDR on the original movement sequences. The reason is that the (movement direction, distance ratio) pairs that are located near the boundaries of quantization subregions may be assigned different symbols and require a *replace* operation that is not needed in the original sequence comparison. For example, given two movement sequences $M_A = [(0, 0.4), (\frac{\pi}{4}, 0.6)]$, $M_B = [(\frac{\pi}{4}, 0.4), (\frac{\pi}{2}, 0.6)]$ and $\epsilon_{dir} = \frac{\pi}{4}$, $\epsilon_{dis} = 0.125$, the corresponding MPSs are: $MPS_A =$ "*bk*" and $MPS_B =$ "*cl*" (using the same quantization map in Figure 3). EDR between original movement sequences $M_A$ and $M_B$ is 0, whereas the ED between $MPS_A$ and $MPS_B$ is 1. As a consequence, trajectory retrieval using ED will introduce false dismissals, which is not allowed in applications that require high accuracy. Therefore, we define a Edit Distance on MPSs (EDM), which is a lower bound of the EDR between original movement sequences. We need to define the cases that two symbols *approximately match*.

**Definition 6.** Two symbols $A_i$ and $A_j$ are said to *approximately match* (denoted by predicate $ap\_match(A_i, A_j)$), if and only if $A_i == A_j$ or $A_i$ is the neighbor of $A_j$. $A_i$ is a neighbor of $A_j$ if the subregions that they represent in the quantization map are directly connected. For example, according to the quantization map in Figure 3, the neighbors of T are K, L, M, S, U, a, b, c and the neighbors of A are H, B, P, I, J (note that movement directions are in polar space).

**Definition 7.** The $EDM(MPS_A(n), MPS_B(m))$ between two MPSs $MPS_A$ and $MPS_B$ of length $n$ and $m$, respectively, is defined as the number of insert, delete, or replace operations that is needed to change $MPS_A$ into $MPS_B$:

$$\begin{cases} n & \text{if } m = 0; \\ m & \text{if } n = 0; \\ EDM(MPS_A(n-1), MPS_B(m-1)) & \\ \qquad \text{if } ap\_match(S_{a,n}, S_{b,m}) & \\ min[EDM(MPS_A(n-1), MPS_B(m-1)) + 1, & \\ \quad EDM(MPS_A(n-1), MPS_B(m)) + 1, & \\ \quad EDM(MPS_A(n), MPS_B(m-1)) + 1] & \\ \qquad \text{otherwise} & \end{cases}$$

The computation of EDM on MPS is the same as that of EDR in Definition 4. Now we need to prove that this EDM on MPS is a lower bound of EDR on original movement sequences.

**Lemma 1.** Given two movement sequences $M_A$ and $M_B$ and their corresponding MPSs $MPS_A$ and $MPS_B$, let $(\theta_{a,i}, \sigma_{a,i})$ and $(\theta_{b,j}, \sigma_{b,j})$ be two (movement direction, distance ratio) pairs of $M_A$ and $M_B$, respectively, and $S_{a,i}$ and $S_{b,j}$ be their corresponding symbols from $MPS_A$ and $MPS_B$. If $match((\theta_{a,i}, \sigma_{a,i}), (\theta_{b,j}, \sigma_{b,j})) = true$, then $ap\_match$ $(S_{a,i}, S_{b,j}) = true$ and if $ap\_match(S_{a,i}, S_{b,j}) = false$, then $match((\theta_{a,i}, \sigma_{a,i}), (\theta_{b,j}, \sigma_{b,j})) = false$.

**Proof:** During the process of mapping (movement direction, distance ratio) pairs into symbols, there are only three cases that could happen:

1. $S_{a,i}$ and $S_{b,j}$ are the same symbol;
2. $S_{a,i}$ and $S_{b,j}$ are neighbors;
3. $S_{a,i}$ and $S_{b,j}$ are not neighbors;

If $match((\theta_{a,i}, \sigma_{a,i}), (\theta_{b,j}, \sigma_{b,j})) = true$, $S_{a,i}$ and $S_{b,j}$ must be the same symbol or neighbors. Otherwise there are at least one subregion gap between them in the quantization map. From the definition of movement pattern alphabet, the size of each subregion is $\epsilon_{dir} * \epsilon_{dis}$. Thus, we have: $|\theta_{a,i} - \theta_{b,j}| > \epsilon_{dir}$ and $|\sigma_{a,i} - \sigma_{b,j}| > \epsilon_{dis}$, which contradict to $match((\theta_{a,i}, \sigma_{a,i}), (\theta_{b,j}, \sigma_{b,j})) = true$. Therefore, according to Definition 6, $ap\_match(S_{a,i}, S_{b,j}) = true$.

If $ap\_match(S_{a,i}, S_{b,j}) = false$, then $S_{a,i}$ and $S_{b,j}$ are not the same symbol or neighbors. As above, we have: $|\theta_{a,i} - \theta_{b,j}| > \epsilon_{dir}$ and $|\sigma_{a,i} - \sigma_{b,j}| > \epsilon_{dis}$. According to Definition 3, $match((\theta_{a,i}, \sigma_{a,i}), (\theta_{b,j}, \sigma_{b,j})) = false$. □

**Theorem 1.** Given two movement sequences $M_A$ and $M_B$, and their corresponding MPSs $MPS_A$ and $MPS_B$, $EDM(MPS_A, MPS_B) \leq EDR(M_A, M_B)$.

**Proof:**

Let $\#ap\_match$ denote the number of pairs of symbols in $MPS_A$ and $MPS_B$ that approximately match (i.e., $\#ap\_match$ $= count(ap\_match(S_{a,i}, S_{b,j}) = true)$, $S_{a,i} \in MPS_A$, $S_{b,j} \in MPS_B$).

Let $\#match$ denote the number of (movement direction, movement distance ratio) pairs in $M_A$ and $M_B$ that match (i.e.,$\#match = count(match((\theta_{a,i}, \sigma_{a,i}), (\theta_{b,j}, \sigma_{b,j})) = true)$, $(\theta_{a,i}, \sigma_{a,i}) \in M_A$, $(\theta_{b,j}, \sigma_{b,j}) \in M_B$).

Define $\overline{\#ap\_match}$ and $\overline{\#match}$ as converses of these (i.e., count where predicates are false). Then, according to Lemma 1, $\#ap\_match \geq \#match$ and $\overline{\#ap\_match} \leq \overline{\#match}$. Therefore the cost (in terms of number of edit operations) of changing $MPS_A$ to $MPS_B$ is less than that of changing $M_A$ to $M_B$. □

Converting movement sequences into MPS has several advantages:

1. MPSs require much less storage space compared to the original movement sequences. For example, if each

```
Procedure MPS1-NN($MQ, MPS_Q, QM, result$) {
  /* $M_Q \equiv$ a query movement sequence; $MPS_Q \equiv$ the query MPS;
     $QM \equiv$ a quantization map; $result \equiv$ the 1-NN movement sequence */
  (1) $smallest\_distance = maxDistance$ /* the 1-NN distance so far */
  (2) for each MPS $MPS_i$ in the database {
  (3)     compute $EDM(MPS_i, MPS_Q)$
  (4)       if ($EDM(MPS_i, MPS_Q) < smallest\_distance$) {
                /* need to check */
  (5)       compute $EDR(M_i, M_Q)$ /* compute true EDR distance */
  (6)         if ($EDR(M_i, M_Q) < smallest\_distance$) { /* update $result$ */
  (7)             $smallest\_distance = EDR(M_i, M_Q)$
  (8)             $result = M_i$
              } /* end-if, line 6 */
          } /* end-if, line 4 */
      } /* end-for, line 2 */
  (9) return result
}
```

**Figure 5: The algorithm for answering 1-NN query using MPS**

symbol in a MPS is stored as a character, a MPS only needs $\frac{1byte}{8bytes} = 12.5\%$ of the storage space needed to store the original movement sequence (assuming a character type needs 1 byte and floating type (real value) needs 4 bytes). As a consequence, the retrieval cost of a MPS should be less than its corresponding movement sequence.

2. According to Theorem 1, the EDM of the two MPSs is a lower bound of the EDR of the original sequences, which guarantees that no false dismissals will be introduced when we use MPS in answering queries such as $k$-nearest neighbors. As mentioned in Section 1, two factors affect the efficiency of retrieval: I/O cost and CPU cost. The MPS representation is based on the assumption that the retrieval cost may be reduced due to the smaller size of MPS compared to movement sequences. Procedure MPS1-NN shown in Figure 5 describes how MPS is used to answer 1-nearest neighbor query (1-NN).
3. Compared to movement sequences, MPSs are one dimensional strings, so the dimensionality reduction techniques for strings can be applied to them. This is addressed in detail in the next section.

## 4. MODIFIED FREQUENCY DISTANCES

Even though we reduce the storage requirements by converting movement sequences into MPSs, the cost of computing the EDM between two MPSs is still $O(n * m)$, since the length of a movement sequence and that of its corresponding MPS are the same (there is a little time saving in terms of comparing each element of the sequences, MPS only compares one symbol and a movement sequence compares two real values). Therefore, directly using MPSs in trajectory retrieval will not reduce that much the computation cost. In [10], a transformation of strings into a multidimensional integer space was proposed by mapping strings to their *frequency vectors* (FV). A frequency vector of a string over an alphabet records the frequency of occurrence of each character of the alphabet in that string. They prove that the *frequency distance* (FD) between the FVs of two strings is a lower bound of the actual edit distance. FD of two points $u$ and $v$ in s-dimensional space, $FD(u, v)$, is defined as the minimum number of steps that is required to go from $u$ to $v$ (or equivalently from $v$ to $u$) by moving to a *neighbor* point at each step. $u$ and $v$ are neighbors if one of them can be

```
Procedure ComputMFD($u, v, QM, result$) {
  /* $u, v \equiv$ s dimensional integer points; $QM \equiv$ a quantization map;
     $result \equiv$ a MFD */
  (1) posDist=0, negDist=0
  (2) for $i = 1$ to $s$ { /*checking moves to its neighbour */
  (3)     $u_i = u_i - v_i$
      } /*end-for line2 */
  (4) for $i = 1$ to $s$ {
  (5)     if ($u_i \neq 0$) {
  (6)         for each neighbor $u_j$ of $u_i$ { /* check each neighbour */
  (7)             if ($u_i * u_j < 0$) {
  (8)                 if ($abs(u_i) > abs(u_j)$) {
  (9)                     $u_i = u_i + u_j$, $u_j = 0$ }
  (10)                else {$u_j = u_j + u_i$, $u_i = 0$ }
                  } /*end-if line7 */
              }/*end-for line6 */
          } /*end-if line5 */
    } /*end-for line4 */
  (11) for $i = 1$ to $s$ {
  (12)   if ($u_i > 0$) $posDist+ = u_i$
  (13)       else $negDist+ = (-u_i)$
    } /*end-for line11 */
  (14) $result = (posDist > negDist)?posDist : negDist$
  (15) return $result$
}
```

**Figure 6: The algorithm for computing the MFD**

obtained from the other using a single edit operation. However, their proof uses standard ED [15] between strings. In contrast to EDM of two MPS that is based on the concept of approximately match (Definition 6), ED is computed based on the equality of two symbols. Therefore their results can not be directly applied to MPS. In order to reduce the dimensionality of MPS, we define a *modified frequency distance* (MFD) as an extension of frequency distance.

**Definition 8.** Let $u$ and $v$ be integer points in $s$- dimensional space. The *modified frequency distance* $MFD(u, v)$ between $u$ and $v$ is defined as the minimum number of steps required to go from $u$ to $v$ (or equivalently from $v$ to $u$) by moving to a *next-to-neighbor* point at each step. $u$ and $v$ are next-to-neighbors if one of them can be obtained from the other using a single edit operation of EDR on movement sequences.

Compared to FD, MFD takes boundary cases into consideration, which corresponds to the approximately match concept defined in Definition 6.

**Theorem 2.** If $MPS_A$ and $MPS_B$ are two strings from the movement pattern alphabet $\mathcal{A} = A_1, A_2, \ldots, A_s$, then $MFD(f(MPS_A), f(MPS_B)) \leq EDM(MPS_A, MPS_B)$, where $f(S_i)$ is the $FV$ of string $S_i$.

**Proof:** A straightforward extension of proof of Theorem in [10].

Based on Theorem 1 and Theorem 2, we have:

**Corollary 1.** Given two movement sequences $M_A$ and $M_B$ of length $n$ and $m$ and their corresponding MPSs $MPS_A$ and $MPS_B$, $MFD(f(MPS_A), f(MPS_B)) \leq EDR(M_A, M_B)$.

Corollary 1 proves that the MFD between two FVs of $MPS_A$ and $MPS_B$ is a lower bound of the EDR between the corresponding movement sequences. Therefore, in order to answer queries such as $k$-nearest neighbor queries, instead of directly computing the EDR of movement sequences, we can compute the MFD to prune out false candidates from the database. Most importantly, the computation cost of MFD is linear! Procedure ComputMFD shown in Figure 6 computes MFD between two FVs. The nested loops in the algorithm may suggest that the computation time of MFD is non-linear. However, as the number of neighbors of each integer point in the frequency space is limited (at

```
Procedure FV1-NN($FV_Q, M_Q, QM, result$) {
  /* $M_Q \equiv$ a query movement sequence; $FV_Q \equiv$ the query FV;
     $QM \equiv$ a quantization map; $result \equiv$ the 1-NN movement sequence*/
  (1) $smallest\_distance = maxDistance$ /* the 1-NN distance so far*/
  (2) for each FV $FV_i$ in the database {
  (3)     compute $MFD(FV_i, FV_Q)$
  (4)       if ($MFD(FV_i, FV_Q) < smallest\_distance$) {
              /* need to check */
  (5)       compute $EDR(M_i, M_Q)$ /* compute true EDR distance */
  (6)         if ($EDR(M_i, M_Q) < smallest\_distance$) {
  (7)             $smallest\_distance = EDR(M_i, M_Q)$
  (8)             $result = M_i$
              } /* end-if, line 6 */
          } /* end-if, line 4 */
      } /* end-for, line 2 */
  (9) return result
}
```

**Figure 7: The algorithm for answering 1-NN query using FV**

most 8), the computation time of Procedure ComputMFD is still linear. Our experimental results also confirm this. Using FV as filter further reduces the retrieval cost since the dimensionality of FV is usually smaller compared to that of movement sequences. Procedure FV1-NN shown in Figure 7 lists steps in answering a 1-NN query with FV as a filter.

## 5. EXPERIMENTS

In this section, we present the experimental results on comparing the efficacy using MPS and the retrieval efficiency using frequency vectors. All experiments were run on a Sun-Blade-1000 workstation with 1G memory under Solaris 2.8. Since $\epsilon_{dir}$ and $\epsilon_{dis}$ are data and application dependent [4, 19], in our experiments, we run several probing 1-NN queries on each data set with different matching thresholds and choose the one that ranks the results close to human observations.

### 5.1 Efficacy of MPSs

As we mentioned in Section 3, converting movement sequences into MPSs has several advantages. However, quantizing real values to symbols will loose accuracy. In this section, we show that using MPSs in trajectory clustering and classification is nearly as effective as using the real movement sequences. In our first experiment, we perform hierarchy clustering using MPSs and movement sequences of two labelled data sets.

The two labelled data sets are generated using Australian Sign Language (ASL) and "cameramouse" (CM) [7] data sets as seeds, respectively. The original CM data set contains 15 trajectories of 5 words (3 for each word) obtained by tracking the tip of finger from videos that record people "write" various words. The original ASL data set from UCI KDD data archive [3] consists of samples of Australian Sign Language signs. 95 signs are collected for 5 different writers. We extract 5 recording for each of following 10 words: "Norway", "cold", "crazy", "eat", "forget", "happy", "innocent", "later", "lose", and "spend", which were also used in [13, 19]. The average lengths of CM and ASL data are 1103 and 65, respectively. We use the program [20] to add interpolated Gaussian noise (about 10-15% of the length of trajectories) and time warping [20] to two data sets. The data set that is generated from CM contains 5 classes and 30 examples of each class, and the one from ASL data contains 10 classes where each class has 50 examples.

[3]http://kdd.ics.uci.edu

The raw trajectories in both data sets are converted into movement direction and distance sequences. For each data set, we take all possible pairs of words (10 pairs for CM data and 45 pairs for ASL data) and use the "complete linkage" hierarchy clustering algorithm [9], which produces the best clustering results [19] to partition them into two clusters. We draw the dendrogram of each clustered result to see whether it correctly partitions the trajectories. The results are reported in Table 1. The Table 1 shows the clustering results using MPS is very close to that of using movement sequences, therefore, we do not loose much accuracy on using MPSs.

| Correct clustering results | movement sequences | MPSs |
|---|---|---|
| CM (total 10 correct) | 10 | 10 |
| ASL (total 45 correct) | 21 | 19 |

**Table 1: Clustering results of using movement sequences and MPSs**

In our second experiment, we carry out simple classification using 1-Nearest Neighbor and test the classification results using the "leave one out" verification mechanism [12]. The error rate is the ratio of number of wrong classifications to the total number of trajectories in the data set. We use the same data sets of Experiment 1. Table 2 reports the results. The classification results using MPSs are a little worse than those using movement sequences.

| Error Rate (%) | movement sequences | MPSs |
|---|---|---|
| CM | 3 | 4 |
| ASL | 9 | 12 |

**Table 2: Classification results of using movement sequences and MPSs**

To summarize, in terms of efficacy, using MPSs is almost same as using original movement sequences. In fact, MPS provides an approximation of the original movement sequences. The accuracy of the approximation can be improved by increasing the number of quantization regions.

### 5.2 Efficiency of FV in Retrieval

As shown in Procedure FV1-NN, FV can be used as a filter to remove false candidates before computing EDM on the movement pattern strings. The aim of the filtering is to remove as many false candidates as possible to reduce the retrieval and computation cost. Therefore, in our experiment, we use *pruning power* as a measure of the filtering effect of FV. The pruning power is measured by $P$, which is defined as the fraction of the data set that must be examined before we can guarantee that the answer to 1-NN is found [11].

$$P = \frac{\text{number of movement sequence that needed to compute EDR}}{\text{total number of movement sequences in the data set}}$$

In the third experiment, we use five data sets with different trajectory lengths. The first two data sets are the same data sets that used in Experiment 1. The third and fourth data sets are synthetic trajectories that are generated by the program used in [22]. The first synthetic data set (SYN1) contains 500 trajectories and the length of each trajectory is 128. The second synthetic data set (SYN2) has 500 trajectories and the length of each trajectory is 512. The program generates the trajectories by simulating the change in moving speeds and directions when people walk freely on a plane. The frequencies of the changes in these values are altered randomly in order to generate complex moving shapes. The trajectories are limited to an area of fixed size (500x500) and the movement directions follow a uniform distribution. The last data set contains 589 trajectories of hockey players, which were extracted from NHL videos. The length of each trajectory is 256.

For each data set, we randomly select 10% of the sequences as query data. A randomly selected movement sequence and FV are used to conduct a 1-NN query using Procedure FV1-NN. Results are averaged over the queries. Figure 8 reports the pruning power of FV on different data sets. The $x$-axis is used to denote different data sets that are arranged from left to right according to their average data length.

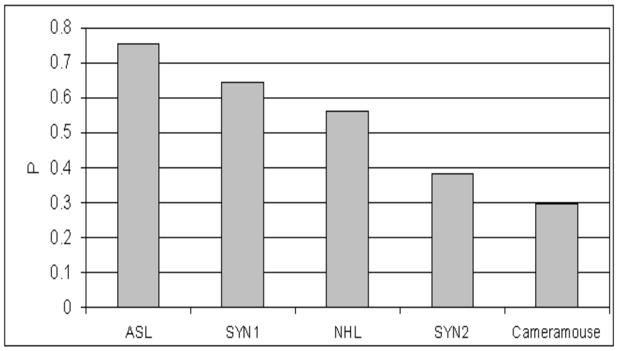


**Figure 8: The pruning power of FV using data sets with different lengths**

As shown in Figure 8, the pruning power of FV drops with increase of the trajectory length. The reason is that FV does not keep the sequence order information of the original movement sequence (only count the frequency).

We conduct the fourth experiment to check the retrieval efficiency of FV in terms of total time that is spent on answering 1-NN queries. Here the total time refers to the time that is spent on data retrieval and the time that is spent on computing the distance measures. Because the real data set we obtained is quite small, we use the same program as in the third experiment to generate 5 synthetic data sets with trajectory lengths ranging from 64 to 1024. Each data set contains 10,000 trajectories. As in the third experiment, we randomly select 10% of the sequences as query data and conduct 1-NN queries using FV (fv-scan). We also run the queries using *linear-scan*, which sequentially scans the movement sequences and computes EDR. All the results are averaged over 50 queries. Figure 9 shows the total time of different methods on different data sets.

As shown in Figure 9, since the computation cost of MFD is only linear, fv-scan needs much less time compared to linear scan. Especially when the trajectory length is shorter, the higher pruning power of FV results in bigger improvements over linear-scan.

In the last experiment, we test the scalability of fv-scan with different sizes of data sets. We run 1-NN queries using fv-scan and linear-scan on 5 synthetic data sets with sizes ranging from 1,000 to 100,000. The trajectory length of all data sets is 256. The results are shown in Figure 10.

In Figure 10, we can see that fv-scan scales smoothly with the increasing database size and it always takes nearly one third of the total time of linear-scan (note that the y-axis scale is logarithmic).

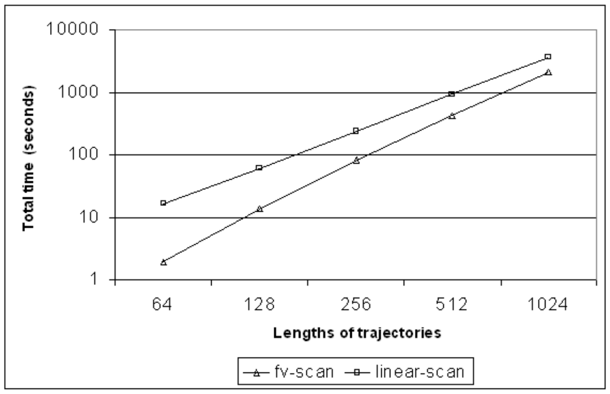


**Figure 9: Comparison of total time of two methods on synthetic data sets with different lengths**

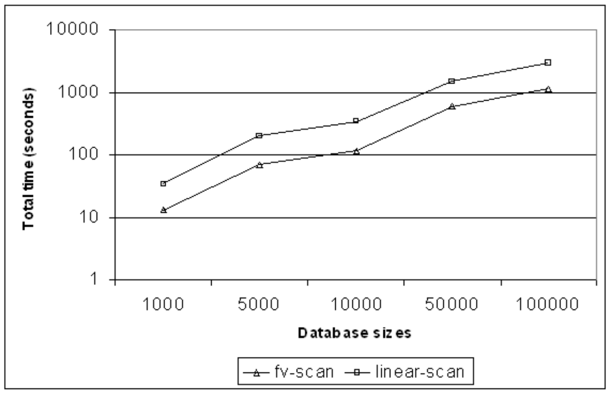


**Figure 10: Comparison of total time of fv-scan and linear-scan on synthetic data sets with different sizes**

Based on the experiments on retrieval efficiency of FV, we draw the following conclusions:

1. In terms of total retrieval efficiency, fv-scan is much better than linear scan due to the linearity of the computation cost of FV. Using FV as a filter can save nearly 2-10 (trajectory length ranges from 60-1000) times of the retrieval cost compared to that of linear scan without the filter.
2. The pruning power of FV scales well with the increasing of database size.

## 6. RELATED WORK

Very limited work has been done on symbolic representation of moving object trajectories. Wai and Chen [21] encode movement directions of moving objects in videos into strings. Li et al. [16] represent the trajectory of a moving object as a sequence of eight movement directions, North (NT), Northwest (NW), Northeast (NE), West (WT), Southwest (SW), East (ET), Southeast (SE), and South (ST). Both approaches only consider the directional and topological information of moving object trajectories and discard the information on movement distances. Several approaches have been proposed to represent one dimensional time series data in symbolic form. Agrawal et al. [1] proposed $\mathcal{SDL}$, which was a language for describing and retrieving the "shape" of one dimensional time series. The "shape" was defined based on the difference of every two consecutive values, which was quantized and represented by a distinct symbol. Huang and

Yu [8] proposed IMPACT algorithm to transform time series data into symbol strings using change ratios between consecutive values. Lin et al. [17] proposed a symbolic representation of one dimensional time series data by first transforming it into piecewise aggregate approximation. They defined a new distance function between the symbolic representation which is a lower bound of the Euclidean distance between original time series data.

Compared to the previous work on symbolic representation, our approach focuses on two dimensional trajectory data. MPSs are obtained from sequences of (movement direction, distance ratio), which encode both movement direction and distance into strings and are invariant to spatial scaling, rotation and translation. Most importantly, we prove that the EDM computed on MPS is a lower bound of the EDR that is computed on original movement sequences, which guarantees that no false dismissals will be introduced during the retrieval. Furthermore, we define MFD between two frequency vectors and use frequency vectors as filters to save the cost of CPU time on computing EDR.

## 7. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel symbolic representation, called movement pattern string (MPS), for moving object trajectories. MPS approximates the real trajectories according to quantization map defined on the space and replaces the pair (movement direction, movement distance ration) by a single symbol. MPS is invariant to spatial scaling, rotation and translation and needs less storage space compared to raw trajectories. Most importantly, we prove that the EDM computed on MPS is a lower bound of the EDR that is computed on original trajectories. Furthermore, we define the modified frequency distance (MFD) between two frequency vectors extracted from MPSs and prove that MFD is also a lower bound of EDR between original sequences. This results means that, during the similarity-based retrieval, we can use the frequency vector as a filter to prune out the false candidates before we compute the EDR between movement sequences. Our experimental results confirm that using MPSs is almost as effective as using raw trajectories and feature vectors with MFD can effectively reduce the false candidates in trajectory retrieval.

Our future work will include finding an embedding method, which keeps both the lower bound property and the temporal order of elements in the strings and investigating the mechanisms to handle sub-trajectory matching efficiently. Furthermore the filtering property of MFD can be used to define some indexes on the movement pattern strings.

## Acknowledgements

Thanks to Michalis Vlachos, Yutaka Yanagisawa, and Eamonn Keogh for providing their source codes or data sets. This research is funded by Intelligent Robotics and Information Systems (IRIS), a Network of Center of Excellence of the Government of Canada.

## 8. REFERENCES

[1] R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zaït. Querying shapes of histories. In *Proc. 21th Int. Conf. on Very Large Data Bases*, pages 502–514, 1995.

[2] Y. Cai and R. Ng. Indexing spatio-temporal trajectoires with chebyshev polynomials. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 599–610, 2004.

[3] L. Chen and R. Ng. On the marriage of edit distance and lp norms. In *Proc. 30th Int. Conf. on Very Large Data Bases.*

[4] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *CS Tech. Report. CS-2004-33, School of Computer Science, University of Waterloo.*

[5] S.-C. Chen and R. L. Kashyap. A spatio temporal semantic model for multimedia presentations and multimedia database systems. *IEEE Trans. Knowledge and Data Eng.*, 13(4):607–622, 2001.

[6] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 419–429, 1994.

[7] J. Gips, M. Betke, and P. Fleming. The camera mouse: Preliminary invertigation of automated visaul tracking for computer access. In *In Proc. Conf. on Rehabilitation Engineering and Assistive Technology Society of North America*, pages 98–100, 2000.

[8] Y. Huang and P. S. Yu. Adaptive query processing for time-series data. In *Proc. 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 282–286, 1999.

[9] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, 1988.

[10] T. Kahveci and A. Singh. Variable length queries for time series data. In *Proc. 17th Int. Conf. on Data Engineering*, pages 273–282, 2001.

[11] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 151–162, 2001.

[12] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. In *Proc. 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 102–111, 2002.

[13] E. Keogh and M. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 285–289, 2000.

[14] S.-L. Lee, S.-J. Chun, D.-H. Kim, J.-H. Lee, and C.-W. Chung. Similarity search for multidimensional data sequences. In *Proc. 16th Int. Conf. on Data Engineering*, pages 599–608, 2000.

[15] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1966.

[16] J.Z. Li, M.T. Özsu, and D. Szafron. Modeling of moving objects in a video databas. In *Proc. 4th Int. Conf. on Multimedia and Computing System*, pages 336–343, 1997.

[17] J. Lin, E. Keogh, S. Lonardi, and P. Patel. Finding motifs in time series. In *Proc. 2nd Int. Workshop Temporal Data Mining*, pages 370–377, 2002.

[18] J. L. Little and Z. Gu. Video retrieval by spatial and temporal sturcture of trajectories. In *Proc. 13th Int. Symp. on Storage and Retrieval for Image and Video Databases*, pages 544–553, 2001.

[19] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *Proc. 18th Int. Conf. on Data Engineering*, pages 673 – 684, 2002.

[20] M. Vlachos, J. Lin, E. Keogh, and D. Gunopulos. A wavelet-based anytime algorithm for k-means clustering of time series. In *Proc. Workshop on Clustering High Dimensionality Data and Its Applications*, pages 23–30, 2003.

[21] T. T. Y. Wai and A. L. P. Chen. Retrieving video data via motion tracks of content symbols. In *Proc. 6th Int. Conf. on Information and Knowledge Management*, pages 105–112, 1997.

[22] Y. Yanagisawa, J. Akahani, and T. Satoh. Shape-based similarity query for trajectory of mobile objects. In *Proc. 4th Int. Conf. on Mobile Data Management*, pages 63–77, 2003.