Mining Following Relationships in Movement Data

Zhenhui Li*, Fei Wu* and Margaret C. Crofoot†‡§
*College of Information Sciences and Technology, The Pennsylvania State University
†University of California, Davis, ‡Max Planck Institute for Ornithology, †Smithsonian Tropical Research Institute
{jessieli, fxw133}@ist.psu.edu, mccrofoot@ucdavis.edu

***Abstract*—Movement data have been widely collected from GPS and sensors, allowing us to analyze how moving objects interact in terms of space and time and to learn about the relationships that exist among the objects. In this paper, we investigate an interesting relationship that has not been adequately studied so far: *the following relationship*. Intuitively, a *follower* has similar trajectories as its *leader* but always arrives at a location with some time lag. The challenges in mining the following relationship are: (1) the following time lag is usually unknown and varying; (2) the trajectories of the follower and leader are not identical; and (3) the relationship is subtle and only occurs in a short period of time. In this paper, we propose a simple but practical method that addresses all these challenges. It requires only two intuitive parameters and is able to mine following time intervals between two trajectories in linear time. We conduct comprehensive experiments on both synthetic and real datasets to demonstrate the effectiveness of our method.**

I. Introduction

Advanced positioning technology enables us to collect a huge amount of movement data from people, animals, and vehicles. Growing interest in mining spatiotemporal interactions among individuals has been driven by important applications in human mobility understanding, ecology studies, and homeland security. For example, various measures [25], [23], [5], [4] have been proposed to find the top-k similar trajectories to a query trajectory. Methods have been developed to detect clusters of moving objects [11], [9], [8], [14], [13], [26].

In this paper, we are particularly interested in mining *following relationships* in movement data because detection of such patterns can benefit many real applications. For example, animal scientists study which individual animal leads the group when animals move in order to determine the social hierarchy, whereas police and security officers look suspicious movements of a criminal who is following a victim. Considering the excessive number of tracking records, it would be extremely difficult for people to manually inspect them and find such patterns.

Given the trajectories of two moving objects, we study how to automatically detect the time intervals in which the following relationship occurs. Generally speaking, to mine the following relationship, it suffices to identify time intervals in which an object (the follower) has similar trajectories as another object (the leader), but always arrives at a location with some time lag.[1] However, in practice this is not an easy task at all, due to the following challenges:

- *Challenge 1. The following time lag is usually unknown and varying*. For example, if a coyote follows a wolf for food, sometimes it may arrive 1 minute late and sometimes the lag could be 10 minutes. In Figure 1, we show an illustrative example where r_1 is 11 minutes behind s_1, but then R catches up with S as r_5 is only 3 minutes behind s_3.
- *Challenge 2. The follower may not have exactly the same trajectory as the leader.* As shown in Figure 1, follower R has a different trajectory from S. In reality, the follower may take a shortcut to catch up with the leader. Or, some followers may intentionally avoid taking the same route as the leader. For example, a suspect may take a different path to avoid being noticed by a victim.
- *Challenge 3. The following relationship could be subtle and always happens in a short period of time.* Various relationships, such as moving together, following, and being independent, could happen between two objects at different time periods. For example, a coyote only follows wolves closely when it is hungry. For the remaining time, its movement could be largely independent of the wolves'. In Figure 1, we can see that R follows S only before time 10:20 and moves together with S afterwards. Therefore, it is crucial to differentiate following relationships from other relationships and to find the correct time intervals in which following relationships actually occur.

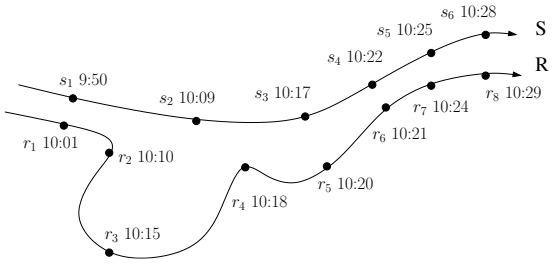


Fig. 1. In this example, object R follows object S from 10:01 to 10:20 and moves together with object S afterwards.

A. Related Work

A limited number of methods have been proposed to detect following relationships in movement data [11], [12], [2], [6]. In the REMO framework [11], [12], [2] and the chasing pattern proposed in [6], a leader should appears in the front region of the follower(s) or move in the same direction. However,

[1] In fact, follower and leader are just two different roles in the same relationship. Our discussion in this paper will be focused on the following relationship, but it can also be applied to the leadership relationship.

the assumptions in these methods are often violated in in real scenarios. In the previous example shown in Figure 1, r_2 is heading downwards at 10:10, and s_2 is apparently not in the front region of r_2.

In the domain of time series research, lagged product-moment correlation (i.e., *cross-correlation*) method has been used as a standard method to capture the time delay in two time series [20], [18], [24]. Similar correlation measures have also been applied to moving objects [21], [15]. In those studies, a window is introduced to capture the local correlation of the trajectories. A major limitation of the windowed correlation measure is that it can only detect relationship with *a constant time lag*. In addition, correlation-based method typically has very high time complexity because it needs to enumerate every time window, time lag, and starting point. For the same reason, as we will see, this method will also *generate a large number of redundant results*. Therefore, the correlation measure is more adequate to determine the significance of dynamic interactions between objects, but it is not suitable for mining such interactions from large-scale data.

B. Our Contribution

In this paper, we first develop a method to solve the following relationship mining problem by transforming the problem into the well-known local sequence alignment (LSA) problem, which has been extensively studied in multiple areas such as bioinformatics and speech recognition. Specifically, LSA aims to detect similar regions in two sequences, such as molecular sequences [22], [1], [17], time series [10], [7], and trajectories [5], by allowing some mismatch, deletion and insertion operations.

For our problem, we define two points (r_i, s_j) as a matching pair if the distance between r_i and s_j is smaller than a threshold $d_{\max}$ and if the time lag between them is smaller than a threshold $l_{\max}$. Then, we use the Smith-Waterman algorithm [22] for LSA to find intervals with most matching pairs. While this method is able to identify some time intervals with following relationships in movement data, we find that its performance is very sensitive to the parameter $d_{\max}$. Take Figure 1 as an example. If $d_{\max}$ is large, the method is likely to incorrectly report the time interval after 10:20 (i.e., when the objects are moving together) as a following interval. Meanwhile, if $d_{\max}$ is small, the true following interval (from time 10:01 to 10:20) will be missed. This example illustrates that LSA solely relies on $d_{\max}$ to differentiate the following relationship from other relationships (see Challenge 3). Unfortunately, an optimal threshold $d_{\max}$ is usually unknown to us, and may not even exist in many real cases.

Motivated by this critical issue of LSA, we further propose a novel method to mine following relationships in movement data. Our key observation is that, if an object R is following S at a certain index i, then for the point r_i on R, there must exist a point s_j on S which is spatially close to r_i, and S visits s_j earlier than R visits r_i. In fact, s_j should be a local minimizer of the distance between r_i and S within a certain time range. For example, in Figure 1 the closest point to r_1 is s_1, and by comparing their timestamps we can see that R is following S at 10:01. More importantly, it is easy to see that the local minimizer *does not change with parameter* $d_{\max}$, which is crucial for the success of our method.

Meanwhile, if there exists a time interval in which R is following S for most of the timestamps, that interval is likely to have a significant following relationship. Therefore, our task becomes finding all such time intervals given two trajectories. To solve this problem efficiently, we show that the problem can be transformed into the well-known Maximum Sum Segment Problem, for which the optimal solution can be found in linear time. In addition, there is no extra parameter needed in the proposed algorithm other than $d_{\max}$ and $l_{\max}$.

We conducted comprehensive experiments on both real and synthetic data to verify the effectiveness of our method. In this paper, we ultimately demonstrate *for the first time* that many interesting and important following relationships can be detected automatically from real animal movement data, despite all the challenges we have mentioned before.

The remainder of the paper is organized as follows. In Section II, we give a formal definition of the following pair. In Section III, we discuss how to find following time intervals by transforming our problem to LSA. Then, in Section IV, we describe in detail our new method for following relationship mining. Experimental studies in Section V demonstrate the effectiveness of our method on both synthetic and real datasets. Finally, we present our conclusions in Section VI.

II. Preliminaries

Suppose we are given the trajectories of two moving objects, denoted by $R = r_1 r_2 \dots r_n$ and $S = s_1 s_2 \dots s_n$ respectively, where r_i and s_i are the locations (i.e., longitude and latitude) recorded at the same timestamp i. For the sake of simplicity, we assume that the locations of moving objects are sampled at synchronized timestamps, though both methods discussed in this paper can easily be extended to handle unsynchronized data.

In order to detect following relationships, we note that if R is following S at timestamp i, then r_i must be spatially close to some location s_j, and S arrives at location s_j ahead of R (i.e., $j < i$). In the definition below, we introduce two parameters $d_{\max}$ and $l_{\max}$ to formally describe the concepts of"spatially close" and "arrives ahead of".

Definition 1 (Following Pair)**.** *Given thresholds* $d_{\max}$ *and* $l_{\max}$*, a location pair* (r_i, s_j) *is said to be a following pair if* $\|r_i - s_j\| < d_{\max}$ *and* $0 < i - j \leq l_{\max}$.

Example 1. *(The running example) We use the example in Figure 2 as a running example in this paper. Here, a green line (solid or dashed) connecting* r_i *and* s_j *means that these two locations are within the distance threshold* $d_{\max}$*. A solid green line in Figure 2 further indicates that* (r_i, s_j) *is a following pair according to Definition 1. In this example,* R *has a following relationship with* S *in time interval [3:11], although* R *is not strictly following* S *at every timestamp in*

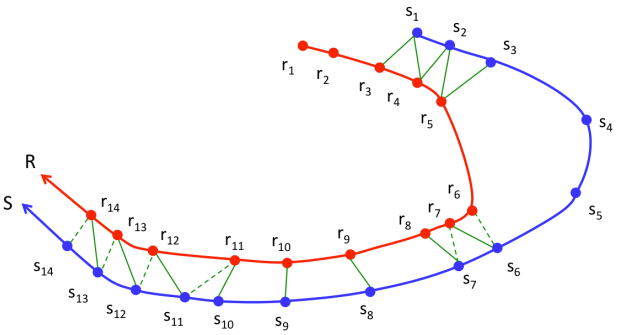


Fig. 2. The running example. R (blue) follows S (red) in time interval [3:11].

this interval. In particular, R takes a shortcut to catch up with S at location r_6. In addition, R starts to move together with S at timestamp 12.

III. Method via Local Sequence Alignment

To mine following relationships from trajectories, a straightforward solution could be mapping our problem onto the local sequence alignment (LSA) problem. The classic LSA problem aims to identify similar subsequences in two symbolic sequences, and can be solved efficiently using a well-known dynamic programming algorithm called the Smith-Waterman algorithm [22]. The algorithm is guaranteed to find the optimal local alignment with respect to the scoring schema. We first briefly introduce the algorithm and then discuss how to use it to find following relationships.

Given two sequences $A = a_1 a_2 \dots a_n$ and $B = b_1 b_2 \dots b_n$ over the alphabet Σ, we define the matching score between any pair (a_i, b_j), where $a_i, b_j \in \Sigma \bigcup \{-\}$, as follows:

$$w(a_i, b_j) = \begin{cases} w(\text{match}) & \text{if } a_i = b_j \\ w(\text{mismatch}) & \text{if } a_i \neq b_j \\ w(\text{gap}) & \text{if } a_i = - \text{ or } b_j = - \end{cases} \quad (1)$$

Here, the symbol "$-$" denotes a gap (insertion or deletion) in the alignment. Then, we use dynamic programming to compute the optimal alignment scores of all subsequences $a_1 a_2 \dots a_i$ $(1 \leq i \leq n)$ and $b_1 b_2 \dots b_j$ $(1 \leq j \leq n)$ and store the scores in matrix H:

$$H(i,j) = \max \begin{cases} 0 & \\ H(i-1, j-1) + w(a_i, b_j) & \text{match/mismatch} \\ H(i-1, j) + w(a_i, -) & \text{deletion} \\ H(i, j-1) + w(-, b_j) & \text{insertion} \end{cases}$$

with $H(i,0) = H(0,j) = 0, 0 \leq i, j \leq n$.

After obtaining the matrix H, we can get the optimally aligned subsequences between A and B. Suppose $H(q, v)$ is the cell with the highest value in matrix H. By tracing back the values of matrix H, we can identify the best aligned subsequences, denoted as $a_p a_{p+1} \dots a_q$ and $b_u b_{u+1} \dots b_v$. To find all the subsequences in A that match with B, we can recursively break sequence A into two subsequences $A_1 = a_1 a_2 \dots a_{p-1}$ and $A_2 = a_{q+1} a_{q+2} \dots a_n$ and find the best alignment between each A_i and B.

It is straightforward to extend the Smith-Waterman algorithm to detect following intervals in trajectories. We only need to modify the matching score in Eq. (1) so that two locations r_i and s_j are considered as a match if and only if (r_i, s_j) is a following pair according to Definition 1. Meanwhile, we note that the Smith-Waterman algorithm has also been applied to trajectory data in [5], which aims to measure the similarity between two trajectories. The previous work [5] has demonstrated that, by quantizing the distances between two locations using a threshold $d_{\max}$, their method is more robust to outliers than other methods using Euclidean distance or dynamic time warping (DTW). Applying the Smith-Waterman algorithm on our problem can effectively address the issue of varying time lags in the following relationship. However, the algorithm is sensitive to parameter d_{max}. We will use Example 2 below to illustrate this flaw.

Example 2. *For the running example in Figure 2, we set $w(\text{match}) = 1$, $w(\text{gap}) = w(\text{mismatch}) = -1$ and show the optimal alignment obtained by using the Smith-Waterman algorithm in Table I. Each column of Table I shows one alignment and its corresponding cost of operation. A gap "$-$" in the row of R or S represents a deletion/insertion operation with penalty -1. A mismatch operation has a penalty score as -1, such as the case for pair (r_6, s_5). The remaining columns with $w = 1$ are the matching cases.*

TABLE I
An optimal sequence alignment.

R	r_1	r_2	r_3	r_4	r_5	-	r_6	r_7
S	-	-	s_1	s_2	s_3	s_4	s_5	s_6
w	-1	-1	1	1	1	-1	-1	1
R	r_8	r_9	r_{10}	r_{11}	r_{12}	r_{13}	r_{14}	-
S	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}
w	1	1	1	1	1	1	1	-1

The corresponding highest cell value is $H(14, 13) = 9$, which is achieved by matching $r_3 \dots r_{14}$ with $s_1 \dots s_{13}$. However, as we can see from Figure 2, R and S are actually moving together during time interval [12:14]. Since the goal of LSA is to maximize the alignment score, the output of LSA will incorrectly report that R is following S during this time interval. We could set $d_{\max}$ to a smaller value so that (r_{12}, s_{11}), (r_{13}, s_{12}), and (r_{14}, s_{13}) are no longer following pairs. But, in this case, the algorithm would fail to detect the true following time interval [3:5] since the distances between r_3, r_4, r_5 and s_1, s_2, s_3 would be larger than $d_{\max}$.

We can see from the above example that, despite its computational efficiency and ability to handle varying time lag, applying the Smith-Waterman algorithm to our problem is flawed in that it solely relies on the distance threshold $d_{\max}$ to differentiate true following relationships from "simply being spatially close". In other words, LSA is very sensitive to the choice of $d_{\max}$. If d_{max} is loose, then this method will incorrectly treat time intervals in which two objects are spatially close as following intervals. If d_{max} is tight, some true following intervals might be lost. In order to overcome

these pitfalls, in the next section we propose a new following relationship mining method that is simple and efficient, yet effective and insensitive to the choice of $d_{\max}$.

IV. Method using Local Distance Minimizer

In this section, we present a novel method to mine following relationships in movement data. We observed that, if R is following S at timestamp i, then there must exist a *strictly positive* integer $\Delta(i)$ such that r_i is spatially close to $s_{i-\Delta(i)}$. In fact, the distance between r_i and S should be minimized *locally* at such $\Delta(i)$. Therefore, we formally define the *following pattern* based on the idea of local distance minimizers as follows.

Definition 2 (Local Minimizer)**.** *Given a time range $I = \{t \in \mathbb{Z}, -l_{\max} \leq t \leq l_{\max}\}$, we define the local minimizer $\Delta(i)$ as*

$$\Delta(i) = \arg\min_{t \in I} \|r_i - s_{i-t}\|. \tag{2}$$

We say a trajectory R is following S at timestamp i (or a following pattern occurs at timestamp i) if $\Delta(i) > 0$ and $\|r_i - s_{i-\Delta(i)}\| < d_{\max}$.

Before proceeding, we make a few important comments about parameters l_{max} and d_{max} and constraint $\Delta(i)$ in Definition 2.

1) Unlike existing methods, which assume a *constant and/or known* time lag, our definition of the following pattern assumes that the time lag is unknown and varying within a range $(0, l_{\max}]$. As we will see later, this relaxation is the key to the successful discovery of subtle following relationships in real movement data.
2) In Definition 2, we assume that the two objects must be within a distance threshold $d_{\max}$ for the following pattern to occur. This is because, in the real world, moving objects cannot have any physical interaction if they are too far away from one another. Consequently, $d_{\max}$ is usually set in a relaxed fashion here to remove timestamps with no interaction. The use of $d_{\max}$ in Definition 2 is different from the way $d_{\max}$ is used in LSA, where it serves to differentiate the following relationship from other relationships (a much more challenging task). Furthermore, we note that for any r_i, the choice of $d_{\max}$ has no effect on the local minimizer $\Delta(i)$. This contrasts with LSA, whose performance relies heavily on $d_{\max}$.
3) We can define the *leading pattern* between R and S in the same way as Definition 2, except that $\Delta(i) < 0$ must hold.

We can find all the timestamps at which R is following S using Definition 2. However, in practice, people are usually more interested in mining *long time intervals with significant following relationships*, since individual following timestamps are less informative in terms of revealing the actual relationship between two objects and are often very sensitive to the noise in movement data. Therefore, in the rest of this section we define the *following score* for any time interval based on Definition 2 and study how to obtain intervals with the highest following scores efficiently.

A. Following Score

Based on Definition 2, we first define a function $f(\cdot)$ at each timestamp as follows:

$$f(i) = \begin{cases} 1, & \text{if } \Delta(i) > 0 \text{ and } \|r_i - s_{i-\Delta(i)}\| < d_{\max} \\ 0, & \text{if } \Delta(i) \leq 0 \text{ and } \|r_i - s_{i-\Delta(i)}\| < d_{\max} \\ \times, & \text{if } \|r_i - s_{i-\Delta(i)}\| \geq d_{\max} \end{cases}$$

As we discussed in the previous section, $f(i) = \times$ indicates that there is no interaction between R and S at timestamp i. Therefore, we say i is *valid* if $f(i) \neq \times$. A time interval I is valid if all the timestamps in I are valid. Note that since the following relationship only occurs in valid time intervals, we can safely break the entire sequence into segments in which all the timestamps are valid and then process each segment independently.[2] Therefore, without the loss of generality, we assume that all the timestamps in the sequence are valid for the rest of this section.

For any time interval I, we further define

$$f(I) = \sum_{i \in I} f(i). \tag{3}$$

We observe that, if two objects are simply moving together in a time interval with no following relationship, we should expect to see *a balanced number* of following and leading patterns for this time interval. Likewise, a time interval with a significant following relationship should *have substantially higher frequency in terms of following patterns compared with the expected number*. Given a time interval I, let $\mu(I)$ denote the number of following patterns which are *expected* to fall into I. Mathematically, $\mu(I)$ is equal to the average frequency observed over all timestamps, multiplied by the length of the interval:

$$\mu(I) = |I| \times Avg\left(f([1:n])\right) = |I| \times \sum_{i=1}^{n} f(i)/n.$$

If there is no following relationship between R and S, then the distribution of $\Delta(i)$ in Definition 2 should be symmetric around 0. In other words, following patterns should occur for about half of the sequence. Hence we have

$$\mu(I) = 0.5 \times |I|. \tag{4}$$

We define the following score as:

$$g(I) = f(I) - \mu(I) = f(I) - 0.5 \times |I|. \tag{5}$$

TABLE II
Example of $f(\cdot)$.

r_i	r_1	r_2	r_3	r_4	r_5	r_6	r_7
nearest point	s_1	s_1	s_1	s_2	s_3	s_6	s_7
$f(i)$	×	×	1	1	1	0	0
r_i	r_8	r_9	r_{10}	r_{11}	r_{12}	r_{13}	r_{14}
nearest point	s_7	s_8	s_9	s_{10}	s_{12}	s_{13}	s_{14}
$f(i)$	1	1	1	1	0	0	0

[2]Throughout the paper, we use the words "interval" and "segment" interchangeably.

Example 3. *Table I shows the value of $f(\cdot)$ as well as the nearest point s_j for each r_i in the running example in Figure 2. We have $f(1) = f(2) =$ ×, because r_1 and r_2 are not close to any points in S (i.e., the distances are larger than $d_{\max}$). Meanwhile, $f(3)$, $f(4)$, $f(5)$, $f(8)$, $f(9)$, $f(10)$, and $f(11)$ take value 1 because the nearest points in S for those points have smaller timestamps (i.e., S arrives at those locations before R). $f(6)$, $f(7)$, $f(12)$, $f(13)$ and $f(14)$ are 0 because S arrives at those locations at the same time or later than R.*

In addition, it is easy to verify that interval $[3:11]$ has the highest following score among all valid intervals, with $g([3:11]) = 7 - 4.5 = 2.5$.

B. Maximizing Following Score

Given the definition of following scores, our goal is to find the intervals with the highest scores.

Problem 1 (Finding Following Time Interval)**.** *Given the thresholds $l_{\max}$ and $d_{\max}$, identify the set of intervals that maximize the following score $g(\cdot)$.*

Next, we show that Problem 1 is equivalent to the well-known Maximum Sum Segments Problem [3], defined as follows:

Problem 2 (Maximum Sum Segments Problem)**.** *Given an input sequence $X = x_1 x_2 \dots x_n$ of real numbers, find K segments with the highest total scores, where the score $\mathcal{F}(X[i:j])$ of a segment $X[i:j] = x_i x_{i+1} \dots x_j$ is equal to the sum of its elements:*

$$\mathcal{F}(X[i:j]) = \sum_{k=i}^{j} x_k. \tag{6}$$

To demonstrate that Problem 1 and 2 are equivalent, it suffices to show that the following score of any given interval is equal to the sum of the following score of the individual timestamps in the interval. The proof is trivial, since we have:

$$g([a:b]) = f([a:b]) - \frac{1}{2}|[a:b]| = \sum_{i=a}^{b} \big(f(i) - \frac{1}{2}\big) = \sum_{i=a}^{b} g([i:i]).$$

Therefore, Problem 1 is now reduced to solving the Maximum Sum Segments Problem with:

$$x_i = g([i:i]) = f(i) - 0.5. \tag{7}$$

However, the standard formulation of the Maximum Sum Segments problem has an obvious disadvantage: a set of heavily overlapping segments may have similar high scores, resulting in redundant detection results. To remedy this issue, we will use the concept of the *maximal segment*:

Definition 3 (Maximal Segment)**.** *Let X be a non-empty score sequence. A segment $X[i:j]$ is maximal in X if (1) all proper sub-segments of $X[i:j]$ have a lower score and (2) no proper super-segments of $X[i:j]$ in X satisfies (1).*

An important property of the maximal segment is that *two maximal sequences cannot overlap* [19]. The proof is intuitive. Basically, given two maximal overlapping sequences, either the union or the intersection of the two has a higher score than one of the two, creating a contradiction. Thus, every element of the input sequence belongs to exactly one maximal segment. Now, we can formally formulate our following relationship mining problem as shown below:

Problem 3 (Finding All Maximal Segments)**.** *Given an input sequence $X = x_1 x_2 \dots x_n$ with $x_i = f(i) - 0.5$, identify the set of all maximal segments of X.*

C. Reverse Test

So far, our analysis has been based on the following patterns from sequence R to S (Definition 2). However, the following relationship is a type of mutual interaction. Therefore, if we assume that R is following S at timestamp i, then S is leading R at i. This observation clearly provides an opportunity to remove false positive segments and to further improve the performance of our method via a reverse test. In this section, we discuss how to modify our following score $g(\cdot)$ to take such information into consideration.

Recall that, given threshold $d_{\max}$ and $l_{\max}$, we say S is *leading* R at timestamp i if $\Delta_r(i) < 0$ and $\|r_i - s_{i-\Delta_r(i)}\| < d_{\max}$, where

$$\Delta_r(i) = \arg \min_{-l_{\max} \le t \le l_{\max}} \|s_i - r_{i-t}\|. \tag{8}$$

Note that here we use the subscript "r" to denote the reverse relationship. Similar to the definition of $f(\cdot)$, we define a new function $f_r(\cdot)$ as follows:

$$f_r(i) = \begin{cases} 1, & \text{if } \Delta_r(i) < 0 \text{ and } \|s_i - r_{i-\Delta_r(i)}\| < d_{\max} \\ 0, & \text{if } \Delta_r(i) \ge 0 \text{ and } \|s_i - r_{i-\Delta_r(i)}\| < d_{\max} \\ \text{×}, & \text{if } \|s_i - r_{i-\Delta_r(i)}\| \ge d_{\max} \end{cases}$$

Now, given any timestamp i at which R is following S ($f(i) = 1$), we can do the reverse test by checking the value of $f_r(i - \Delta(i))$. Based on the result of the reverse test, we define the new following score $g'(\cdot)$ as follows:

$$g'(i) = \begin{cases} 1, & \text{if } f(i) = 1 \text{ and } f_r(i - \Delta(i)) = 1 \\ 0, & \text{if } f(i) = 1 \text{ and } f_r(i - \Delta(i)) = 0 \\ -1, & \text{if } f(i) = 0 \end{cases} \tag{9}$$

Note that $f_r(i - \Delta(i))$ cannot take value -1 when $f(i) = 1$, hence $g'(i)$ is well-defined. Finally, we can state the modified following relationship mining problem as follows.

Problem 4. *Given an input sequence $X = x_1 x_2 \dots x_n$ with $x_i = g'(i)$, identify the set of all maximal segments of X.*

TABLE III
EXAMPLE OF $g'(\cdot)$.

r_i	r_1	r_2	r_3	r_4	r_5	r_6	r_7
$g'(i)$	×	×	1	1	1	−1	−1
r_i	r_8	r_9	r_{10}	r_{11}	r_{12}	r_{13}	r_{14}
$g'(i)$	0	1	1	1	−1	−1	−1

Example 4. *Comparing Table II with Table III, we see that most of the 1's and 0's in $f(i)$ become 1' and -1's, respectively, in $g'(i)$. The only exception is that $g'(8) = 0$, because r_8 is following s_7, but s_7 is not leading r_8. The closest point in R to s_7 is r_7. It can be further shown that interval $[3:11]$ again has the highest score among all valid intervals.*

D. Algorithm

In this section, we describe the general algorithm for finding all the following intervals from a pair of trajectories. We name the specific algorithms that solve Problems 3 and 4 as **FOL-1** and **FOL-2**, respectively.

First we note that, given a pair of trajectories with length n, it takes $O(n \cdot l_{\max})$ time to compute the following scores $g(\cdot)$ or $g'(\cdot)$ for all the timestamps. Since $l_{\max}$ is typically a small constant compared with n for real applications, we see that the time complexity of computing the following scores is linear in n.

Then, given the sequence of following scores X, we use a linear-time algorithm (i.e., $O(n)$) proposed in [19] to find all the maximal segments in X. The pseudo code is shown in Algorithm 1. We refer interested readers to [19] for detailed explanations and proofs.

Based on the above analysis, we conclude that the overall time complexity of our method is $O(n)$.

Algorithm 1 (Find All Maximal Following Time Intervals)

INPUT: A sequence X of real numbers.
OUTPUT: Maximal time intervals.
ALGORITHM:

```
sum ← 0
k ← 0
for i ← 1 to n do
    sum ← sum + x_i
    if x_i > 0 then
        k ← k + 1
        // A new interval I_k with one element x_i
        I_k.s ← i, I_k.t ← i
        L_k ← sum − x_i, R_k ← sum
        while true do
            j is the first L_j searched from L_{k−1} to L_1 that
            L_j < L_k
            if (j does not exist) or (R_j ≥ R_k) then
                break
            // Merge intervals I_j, I_{j+1}, ..., I_k
            I_j.t ← i, R_j ← R_k
            k ← j
Output all intervals: I_1, ..., I_k
```

V. Experiment

In this section, we present a comprehensive performance study of our method on both synthetic and real datasets, and compare it with three other methods: REMO, correlation-based method and LSA.

A. Methods for Comparison

REMO. REMO [2] targets finding leaders among a group of moving objects for k consecutive timestamps. To compare it with our method, we set the size of the potential followers to 1. Given two sequences R and S, the method constructs a binary array storing whether s_i appears in the *front region* of r_i parameterized by an apex angle α, a radius r, and an angle β restricting their difference in direction $\|d_i - d_j\|$. Figure 3 provides an illustration of this method. A following interval is defined as an interval with consecutive 1's in the binary array.

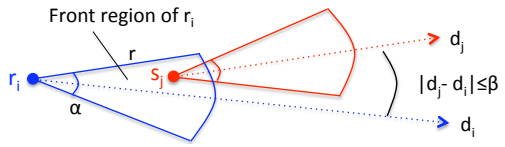


Fig. 3. Front region defined in REMO [2].

Note that REMO is originally designed for identifying leaders in bird migration movement and uses the geometric concept of a "front region" to capture the following relationships. However, in general movement data, a leader does not necessarily appear in the front region of the followers. To maximize REMO's ability to mine general following relationships, we set its parameters to $\alpha = \pi/2$, $\beta = \pi$ and $r = \text{Inf}$. As a result, REMO will report object R follows object S at timestamp i, as long as R is in front of S at i.

Correlation-based method. Cross-correlation method has been used as a standard method to capture the *constant* time delay between two time series [20], [18], [24] and trajectories [21], [15]. In these methods, a window is often used to measure the local correlation between sequences. Given two sequences R and S, let $Corr(i, w, l)$ denote the normalized cross-correlation method between subsequences $s_{i-l} \dots s_{i+w-1-l}$ and $r_i \dots r_{i+w-1}$, where w is the window size and l is the time lag, we have

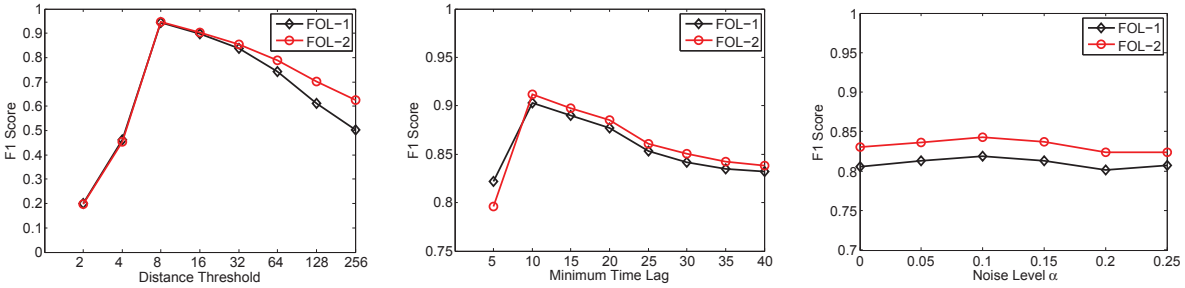
$$Corr(i,w,l) = \frac{\sum_{t=i}^{i+w-1}(r_t - \overline{r}) \cdot (s_{t-l} - \overline{s})}{\sqrt{\sum_{t=i}^{i+w-1}|r_t - \overline{r}|^2}\sqrt{\sum_{t=i}^{i+w-1}|s_{t-l} - \overline{s}|^2}}$$

where $\overline{r} = \frac{1}{w}\sum_{t=i}^{i+w-1} r_t$ and $\overline{s} = \frac{1}{w}\sum_{t=i}^{i+w-1} s_{t-l}$. We enumerate all the possible triplets (i, w, l) and find the set of triplets that maximize $Corr(i, w, l)$.

Local sequence alignment (LSA). As we discussed in Section III, our problem can be cast as an LSA problem and solved efficiently via the Smith-Waterman algorithm. Compared with our method, LSA requires parameters on the weights of three operations: $w(\text{match})$, $w(\text{gap})$ and $w(\text{mismatch})$. We tuned them separately and found that LSA achieves the best performance when $w(\text{match}) = 1$ and $w(\text{gap}) = w(\text{mismatch}) = -1$. Therefore, we use this setting for all the experiments in this paper.

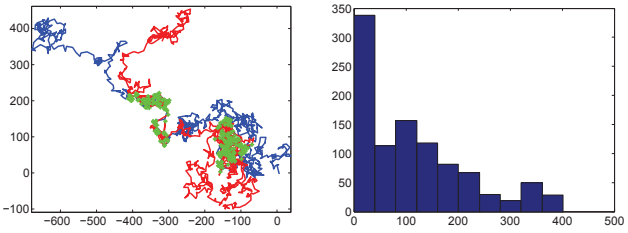
B. Synthetic Dataset Generation

To quantitatively evaluate the performance of all methods, we take the following steps to generate a pair of trajectories



(a) Distance threshold $d_{\max}$ (b) Maximum time lag $l_{\max}$ (c) Noise level α

Fig. 5. Parameter sensitivity.



(a) (b)

Fig. 4. Synthetic sequence example. (a) Two synthetic trajectories (blue and red). Green dots indicate locations where following patterns occur. (b) Histogram of the minimum distances, $\|r_i - s_{i-\Delta(i)}\|$, for all the timestamps.

R and S with a partial following relationship.

1) A random walk sequence $S = s_1 s_2 \dots s_n$ is generated according to the popular *Rayleigh flight* model [16]. At each timestamp, the step size follows a normal distribution $\mathcal{N}(0, \sigma_d^2)$, whereas the direction is random and isotropic.
2) For following intervals, select a set of time intervals $\mathcal{I}^* = \{I_i^* = [a_k : b_k]\}_{k=1}^m$, where $1 \le a_1 < b_1 < a_2 < \dots < b_m \le n$. For each timestamp $i \in \mathcal{I}^*$, we draw a time lag l_i uniformly from $[l_{\min}^*, l_{\max}^*]$, where $l_{\max}^* \ge l_{\min}^* > 0$. We further enforce $|l_{i+1} - l_i| \le 2$ to mimic the practical situation in which the time lag does not change abruptly at consecutive timestamps.
3) Perturb the following interval by changing l_i to $-l_i$ for each $i \in \mathcal{I}^*$ with probability α ($0 \le \alpha < 0.5$).
4) Compose the following segments in R. For each $i \in \mathcal{I}^*$, set $r_i = s_{i-l_i} + d_i$, where d_i is a random variable drawn uniformly from a disk at the origin with radius $d_{\max}^*$.
5) Fill in the remaining segments of R with random walk sequences generated according to the Rayleigh flight model with the same parameters as in Step 1.

Unless otherwise stated, we use the following default values to generate the synthetic sequences: $n = 1000$, $\sigma_d = 8$, $\mathcal{I}^* = \{[100 : 250], [700 : 800]\}$, $l_{\max}^* = 10$, $l_{\min}^* = 1$, $\alpha = 0.1$ and $d_{\max}^* = 8$. Figure 4 shows an example of the sequences generated according to the above procedure.[3]

[3]Note that we set $n = 1000$ simply because the correlation-based method is too slow (i.e., $O(n^4)$) when n is large. Our method has linear time complexity and can handle much larger n.

We evaluate the performance of all methods using F_1 score, a commonly used measure in the field of information retrieval. For our problem, let $\mathcal{I} = \{I_1, I_2, \dots\}$ denote a set of time intervals with a following relationship discovered by any method, we define the precision and recall as follows:

$$precision = \frac{|\bigcup_{\forall i,j}(I_i \bigcap I_j^*)|}{|\bigcup_{\forall i} I_i|}, \quad recall = \frac{|\bigcup_{\forall i,j}(I_i \bigcap I_j^*)|}{|\bigcup_{\forall j} I_j^*|},$$

where $\mathcal{I}^* = \{I_1^*, I_2^*, \dots\}$ is the set of ground truth intervals. Then, the F_1 score is

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}.$$

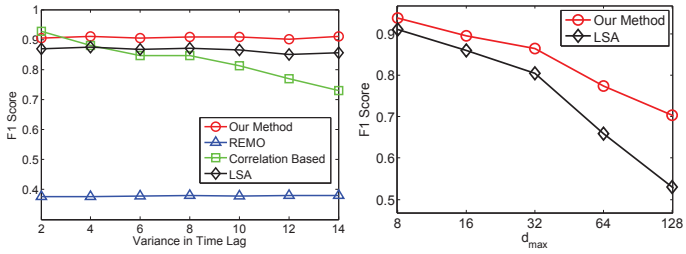
C. Sensitivity of Parameters

In this section, we use synthetic data to study the sensitivity of our methods FOL-1 and FOL-2 w.r.t. the parameters $d_{\max}$, $l_{\max}$, as well as the noise level α. For each setting of the parameters, we report the average F_1 score over 100 trials.

Performance w.r.t. distance threshold $d_{\max}$. In this experiment, we fix $l_{\max} = 15$ and show the F_1 score of our methods under different values of $d_{\max}$ in Figure 5(a). Note that $d_{\max}$ grows exponentially in this experiment. As expected, both FOL-1 and FOL-2 achieve the best performance when $d_{\max} = d_{\max}^* = 8$. Also, with the reverse test, FOL-2 outperforms FOL-1, especially when $d_{\max}$ is large (hence a large number of false positives may occur). In addition, our method does not require the estimated $d_{\max}$ to be close to $d_{\max}^*$, as long as $d_{\max} > d_{\max}^*$. For example, with $d_{\max} = 8 \cdot d_{\max}^* = 64$, FOL-2 still achieves a F_1 score around 0.8. The observation of our method being insensitive to $d_{\max}$ is very important since $d_{\max}^*$ is typically unknown in a real-world setting.

Performance w.r.t. maximum time lag $l_{\max}$. In this experiment, we fix $d_{\max} = 20$ and show the F_1 score of our methods under various $l_{\max}$, as shown in Figure 5(b). Both methods achieve the best performance when $l_{\max} = l_{\max}^* = 10$. More importantly, our methods are not sensitive to the choice of $l_{\max}$, as long as $l_{\max} > l_{\max}^*$.

Performance w.r.t. noise level α. In this experiment, we fix $l_{\max} = 15, d_{\max} = 40$ and show the F_1 score of our method under various α in Figure 5(c). As the noise level increases, the performance of our methods remains roughly the same, because our methods are designed to detect any segment as long as its following score is higher than expected.



(a) (b)

Fig. 6. (a). Comparison of all methods w.r.t variance in time lag. (b). Comparison of our method and LSA w.r.t parameter $d_{\max}$.

In summary, we have demonstrated that *our methods are not sensitive to the parameters* $d_{\max}$ *and* $l_{\max}$*, or to noises in the data*. Since FOL-2 is more robust than FOL-1, we use FOL-2 as our method in the following experiments.

D. Comparison with Other Methods on Synthetic Dataset

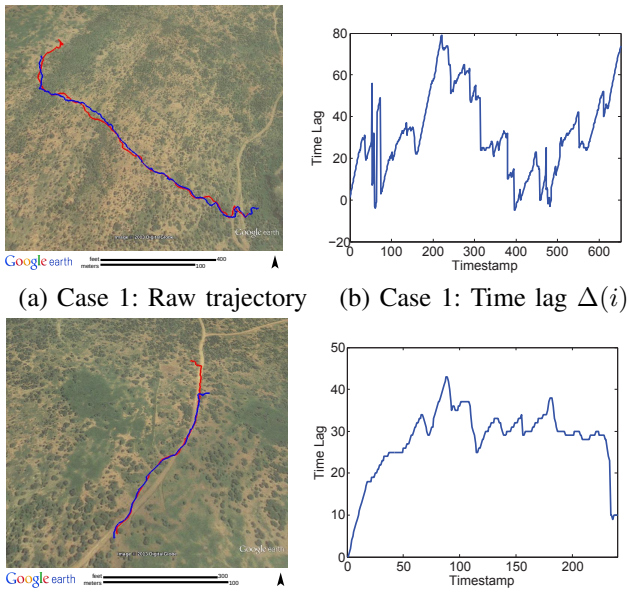
In this section, we first compare all the methods under different levels of time lag variance Δ_l. In this experiment, we set $\mathcal{I}^* = \{[100 : 300]\}$ and create multiple time lag ranges with different variances by setting $[l^*_{\min}, l^*_{\max}] = [l_0 - \Delta_l, l_0 + \Delta_l]$ with $l_0 = 16$. For our method and LSA, we set $d_{\max} = 16, l_{\max} = 30$. We supply the correlation-based method with the ground truth window size ($= 201$) to make it computationally feasible, and we only keep the interval with the highest correlation score as the result. In a real-world scenario, such a ground truth window is usually unknown. Also, the correlation-based method takes 13 seconds with a given time window, while all the other methods take less than 1 second.

In Figure 6(a), we show the F_1 score of all the methods as a function of time lag variance Δ_l. Our method is not affected by the increasing time lag variance, and it outperforms other methods when the variance is large. In contrast, the performance of the correlation-based method quickly degrades as Δ_l increases. Meanwhile, REMO performs poorly in all cases because it requires one object to appear in the front region of the other object for the following pattern to occur, whereas we assume random and isotropic moving direction when generating the synthetic data.

In Figure 6(b), we further examine the sensitivity of our method and LSA to the parameter $d_{\max}$. When $d_{\max}$ increases from $d^*_{\max} = 8$ to 128 (16 times), the F_1 score of LSA drops from 0.9 to 0.52, whereas the F_1 score of our method only drops to 0.7. These results show that $d_{\max}$ affects our method less than it does LSA. Such a difference will become even more obvious on real dataset.

E. Case Study on a Real Dataset

In this section, we conduct a case study and a performance comparison on a real dataset. The real dataset contains GPS locations (longitude and latitude) of a group of 26 baboons tracked from August 1 to August 27, 2012 in Laikipia, Kenya. The sampling rate of this dataset is 1 location per second. For all the experiments, we use $d_{\max} = 50$ (meters) and $l_{\max} = 60$ (seconds) for both our method and LSA.



(a) Case 1: Raw trajectory (b) Case 1: Time lag $\Delta(i)$

(c) Case 2: Raw trajectory (d) Case 2: Time lag $\Delta(i)$

Fig. 7. Two examples of the following relationships detected by our method in real baboon movement data.

We pick two baboons for this case study and name them A and B. In Figure 7, we show two cases of following patterns detected by our method. The first case happens between 8:00AM - 9:00AM on August 5, and the second case happens between 9:00AM - 10:00AM on August 3. In both cases, B follows A. In the first case, the following relationship lasts about 9 minutes; in the second case, it lasts 4 minutes. The trajectories of baboon A (indicated by the red line) and B (indicated by the blue line) are plotted on Google Earth in Figure 7(a) and 7(c). Readers are also encouraged to view the animations online[4] to see the dynamics of the following relationships.

In Figure 7(b) and 7(d), the time lag varies substantially during the following period. Our experiments suggest that, in the real world, the time lag that characterizes the relationship between leaders and followers can be highly variable. The experiment results also demonstrate that our method is able to detect following relationships despite such variation. Next, we compare the performance of our method with other methods using the trajectories of the same two baboons between 10:00AM - 11:00AM on August 2.

Comparison with REMO. In Figure 8, we show the following intervals found by REMO and by our method. REMO reports many short intervals, whereas we only report intervals that are longer than 10 seconds. Note that red bars indicate intervals when B follows A, whereas blue bars denote intervals when A follows B (obtained by reversing the roles of the two objects).

In Figure 8, REMO tends to find many small segments in a long following interval due to its strict definition of front

[4]http://faculty.ist.psu.edu/jessieli/icdm13/following.html

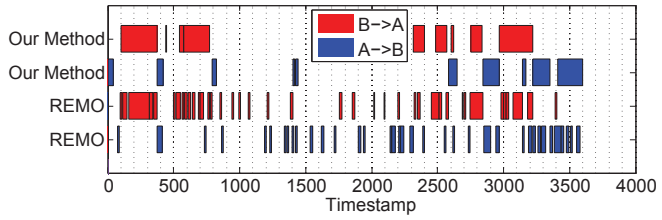


Fig. 8. Following intervals found by our method and by REMO.

region, while our method is able to find all the long following intervals. REMO also reports many small non-following intervals, such as those in the time period $[800 : 2200]$.

In Figure 9(a), we further plot one of the following intervals $[2969 : 3221]$ detected by our method (see animation online). REMO breaks this 253-second following interval into 17 small following intervals. In Figure 9(b)-(f), we show the five intervals detected by REMO with lengths longer than 10 seconds. Even with the relaxed parameters, REMO is not able to detect the entire following range in real movement data. Compared with the long time intervals detected by our method, such small segments detected by REMO clearly carry much less useful information about the animals' movement behaviors and are more difficult for the field experts to interpret.

TABLE IV
TOP-3 INTERVALS OBTAINED BY THE CORRELATION-BASED METHOD WITH VARIOUS WINDOW SIZES.

Window Size	Intervals		
20	[2858:2877]	**[382:401]**	[159:178]
30	[2862:2891]	[260:289]	**[382:411]**
40	**[382:421]**	[2841:2880]	[191:230]
50	**[355:404]**	[2841:2890]	[186:235]
100	**[354:453]**	[159:258]	[2961:3060]
200	**[47:246]**	**[274:473]**	[2706:2905]
300	**[212:511]**	[2372:2671]	[2688:2987]

Comparison with correlation-based method. In Table IV, we show the following intervals detected by the correlation-based method. Since this method has very high computational complexity (i.e., $O(n^4)$), it is not feasible to enumerate all possible values for the parameters on a real dataset. Therefore, in Table IV, we choose a small number of window sizes and report the Top-3 time intervals for each window size.

There are several issues with the results. First, since this method cannot handle varying time lags, the detected time interval typically contains a mixture of following and non-following relationships. For example, in Figure 10 we show the actual time lag of the interval $[354 : 453]$ (detected with window size 100), which fluctuates around 0. Second, it is impossible for the correlation-based method to detect all the following intervals with a single fixed window size. The results obtained by enumerating multiple window sizes are highly redundant (see the highlighted intervals in Table IV). It is not clear how such results can be refined to provide meaningful insight into movement behaviors.

Comparison with LSA. In Figure 11, we compare the intervals found by LSA with those found by our method. When

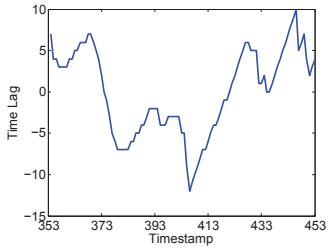


Fig. 10. The actual time lag in interval $[354 : 453]$.

$d_{\max} = 50$ (meters), the results obtained by LSA include many overlapping intervals (i.e., intervals in which both A follows B *and* B follows A), such as $[2969 : 3221]$. However, two objects cannot follow each other at the same time. Indeed, as indicated by the results obtained by using our method, only A is following B in this time interval (see animation online). LSA incorrectly reports that B also follows A because, with a relatively large $d_{\max}$, locations on A can still form many following pairs with locations on B.

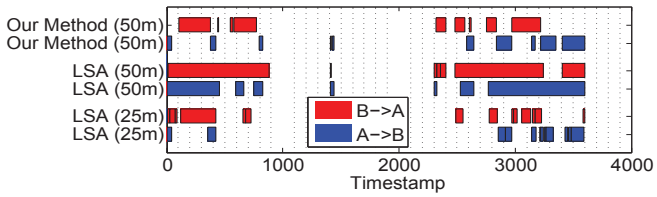


Fig. 11. Following intervals found by our method and by LSA.

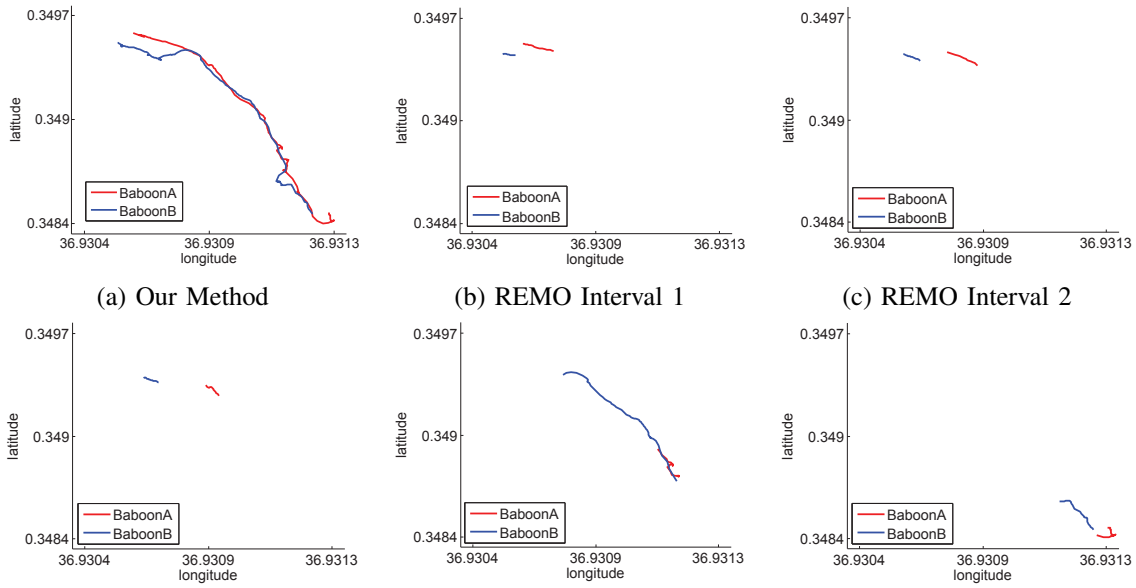
If we reduce $d_{\max}$ to 25 (meters), some overlapping intervals are removed. The results of using LSA are shown in Figure 11. In this case, LSA breaks several long following intervals into smaller ones. In addition, some true following intervals are missed by LSA, such as the red interval between [2300:2400]. From these results, given a fixed value for $d_{\max}$, LSA cannot identify all the long following intervals without reporting overlapping intervals.

In summary, through this case study, we have verified that the challenges we mentioned in Section I indeed exist in real movement data. In addition, ours is the only method that can successfully and efficiently retrieve the interesting following time intervals from real movement data despite the challenges.

VI. CONCLUSION

In this paper, we address an interesting and challenging problem in spatiotemporal data mining: detecting following relationships. Unlike existing methods, which rely on over-strict definitions of following patterns, our method is able to mine general following relationships from real movement data in linear time. Our method addresses challenges including (1) unknown and varying time lag; (2) dynamics in trajectories; and (3) subtle relationships that exist only in a short period of time. Experimental results for both real and synthetic data demonstrate the effectiveness of our method.

While our method has been focused on mining following relationships between two moving objects, one important



(a) Our Method (b) REMO Interval 1 (c) REMO Interval 2

(d) REMO Interval 3 (e) REMO Interval 4 (f) REMO Interval 5

Fig. 9. Case 3: Comparison of our method with REMO on the real baboon dataset.

extension is mining leaders and followers among a large group of moving objects. For example, an animal may lead the migration of its herd, or a person might be followed a gang of suspects. To mine such relationships, we could construct a directed graph on the objects with edges indicating the pairwise following relationships. Then, a node with a high in-degree could be the potential leader of a group. In the future, we plan to develop more efficient algorithms which do not require computing the pairwise relationships.

REFERENCES

[1] S. F. Altschul, M. S. Boguski, W. Gish, and J. C. Wootton. Issues in searching molecular sequence databases. *Nature genetics*, 6(2):119–129, 1994.

[2] M. Andersson, J. Gudmundsson, P. Laube, and T. Wolle. Reporting leaders and followers among trajectories of moving point objects. *GeoInformatica*, 12(4):497–528, 2008.

[3] J. L. Bates and R. L. Constable. Proofs as programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 7(1):113–136, 1985.

[4] L. Chen and R. T. Ng. On the marriage of lp-norms and edit distance. In *Proc. 2004 Int. Conf. Very Large Data Bases (VLDB'04)*, 2004.

[5] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *Proc. 2005 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'05)*, 2005.

[6] F. de Lucca Siqueira and V. Bogorny. Discovering chasing behavior in moving object trajectories. *Transactions in GIS*, 15(5):667–688, 2011.

[7] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

[8] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen. Discovery of convoys in trajectory databases. In *Proc. 2008 Int. Conf. Very Large Data Bases (VLDB'08)*, 2008.

[9] P. Kalnis, N. Mamoulis, and S. Bakiras. On discovering moving clusters in spatio-temporal data. In *Proc. 2005 Int. Symp. Spatial and Temporal Databases (SSTD'05)*, pages 364–381, 2005.

[10] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.

[11] P. Laube and S. Imfeld. Analyzing relative motion within groups of trackable moving point objects. In *Proc. 2002 ACM Int. Symp. Advances in Geographic Information Systems (GIS'02)*, pages 132–144, 2002.

[12] P. Laube, S. Imfeld, and R. Weibel. Discovering relative motion patterns in groups of moving point objects. *International Journal of Geographical Information Science*, 19(6):639–668, 2005.

[13] X. Li, , V. Ceikute, C. Jensen, and K.-L. Tan. Effective online group discovery in trajectory databases. *Knowledge and Data Engineering, IEEE Transactions on*, 2012.

[14] Z. Li, B. Ding, J. Han, and R. Kays. Swarm: Mining relaxed temporal moving object clusters. In *Proc. 2010 Int. Conf. Very Large Data Bases (VLDB'10)*, Singapore, Sept. 2010.

[15] J. Long and T. Nelson. Measuring dynamic interaction in movement data. *Transactions in GIS*, 2012.

[16] B. B. Mandelbrot. *The fractal geometry of nature*. Times Books, 1982.

[17] D. W. Mount. Sequence and genome analysis. *Bioinformatics: Cold Spring Harbour Laboratory Press: Cold Spring Harbour*, 2, 2004.

[18] S. Papadimitriou, J. Sun, and P. Yu. Local correlation tracking in time series. In *Proc. 2006 Int. Conf. Data Mining (ICDM'06)*, pages 456–465, 2006.

[19] W. L. Ruzzo and M. Tompa. A linear time algorithm for finding all maximal scoring subsequences. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 234–241. AAAI Press, 1999.

[20] Y. Sakurai, S. Papadimitriou, and C. Faloutsos. Braid: Stream mining through group lag correlations. In *Proc. 2005 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'05)*, pages 599–610, 2005.

[21] T. Shirabe. Correlation analysis of discrete motions. *Geographic Information Science*, pages 370–382, 2006.

[22] T. F. Smith and M. S. Waterman. Comparison of biosequences. *Advances in Applied Mathematics*, 2(4):482–489, 1981.

[23] M. Vlachos, D. Gunopulos, and G. Kollios. Discovering similar multidimensional trajectories. In *Proc. 2002 Int. Conf. Data Engineering (ICDE'02)*, pages 673–684, San Fransisco, CA, April 2002.

[24] D. Wu, Y. Ke, J. Yu, P. Yu, and L. Chen. Detecting leaders from correlated time series. In *Proc. 2010 Int. Conf. on Database Systems for Advanced Applications (DASFAA'10)*, pages 352–367, 2010.

[25] B.-K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proc. 1998 Int. Conf. Data Engineering (ICDE'98)*, pages 201–208, Orlando, FL, Feb. 1998.

[26] K. Zheng, Y. Zheng, N. J. Yuan, and S. Shang. On discovery of gathering patterns from trajectories. In *Proc. 2013 Int. Conf. Data Engineering (ICDE'13)*, 2013.