Feature Grouping-Based Outlier Detection Upon Streaming Trajectories

Jiali Mao, Tao Wang, Cheqing Jin, *Member, IEEE*, and Aoying Zhou, *Member, IEEE*

Abstract—Outlier detection acts as one of the most important analysis tasks for trajectory stream. In stream scenarios, such properties as unlimitedness, time-varying evolutionary, sparsity, and skewness distribution of trajectories pose new challenges to outlier detection technique. Trajectory outlier detection techniques mainly focus on finding trajectory that is dissimilar to the majority of the others, which is based on the hypothesis that they are probably generated by a different mechanism. Most distance-based methods tend to utilize a function (e.g., weighted linear sum) to measure the similarity of two arbitrary objects provided that representative features have been extracted in advance. However, this kind of method is not tailored to identify the outlier which is close to its neighbors according to some features, but behaves significantly different from its neighbors in terms of the other features. To address this issue, we propose a feature grouping-based mechanism that divides all the features into two groups, where the first group (*Similarity Feature*) is used to find close neighbors and the second group (*Difference Feature*) is used to find outliers within the similar neighborhood. According to the feature differences among local adjacent objects in one or more time intervals, we present two outlier definitions, including local anomaly trajectory fragment (*TF-outlier*) and evolutionary anomaly moving object (*MO-outlier*). We devise a basic solution and then an optimized algorithm to detect both types of outliers. Experimental results show that our proposal is both effective and efficient to detect outliers upon trajectory data streams.

Index Terms—Feature grouping, trajectory stream, local anomaly trajectory fragment, evolutionary anomaly moving object

1 Introduction

With the vigorous development of modern mobile devices and location acquisition technologies, the positions of moving objects are acquired instantaneously and collected continuously in a streaming manner. For instance, the taxis equipped with GPS sensors relay their location information to a central server at intervals like 60 seconds, so that the taxi-company is capable of processing taxi-hailing requests efficiently. Such a data stream is essentially a continuous, infinite sequence of positions accompanied and ordered by explicit timestamps. Effective analysis on streaming trajectories fosters a broad range of critical applications, such as route planning and recommendation [1], [2], [3], [4], intelligent transportation management [5], road infrastructure optimization [6], etc. In this paper we are primarily concerned with outlier detection upon trajectory streams.

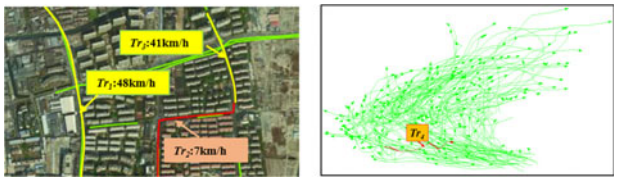
Hawkins intuitively defines outlier as an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism [7]. Barnett states that an outlier is an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data [8]. To some extent, the phrase "deviates from other observations" or "inconsistent with the remainder of that set of data" is subjective and context-based. Trajectory outlier is usually regarded as an abnormal trajectory that is obviously distinct from the majority of trajectories according to a certain similarity criterion, such as the vessels whose movement behaviors are significantly different from others in the same area of ocean [9], the hurricanes that suddenly change wind direction [10], the taxis with detour behaviors [11], [12], and unexpected road changes [13], [14], etc.

Considerable efforts have been invested toward defining an appropriate similarity criterion to measure the closeness among trajectories. First of all, an indispensable step is to extract representative features of each trajectory, based on which a function is utilized to measure the pairwise similarity (or distance) among them. Although the weighted linear sum of the distances for all features is widely adopted as the overall function [10], [15], it is insensitive to the situation where trajectories are similar in some features, but significantly different in the rest.

Fig. 1a illustrates a small example of three trajectories derived by taxis at some time interval, denoted as Tr_1, Tr_2 and Tr_3 respectively. They are close to one another, among which Tr_2 (in red) with speed of $7\ \mathrm{km/h}$ is most likely to be an outlier, because the average speed of its neighbors Tr_1 and Tr_3 is about $44\ \mathrm{km/h}$. Such an outlier may indicate a traffic jam. However, whether it is an outlier or not in the traditional way depends on the weight of speed feature. It won't be identified as an outlier accurately if the weight of

- *J. Mao is with the School of Data Science and Engineering, East China Normal University, Shanghai 200062, China, and the School of Computing, China West Normal University, Nanchong, Sichuan 637009, China. E-mail: jlmao1231@stu.ecnu.edu.cn.*
- *T. Wang is with the School of Computer Science and Software Engineering, East China Normal University, Shanghai 200062, China. E-mail: toy_king@stu.ecnu.edu.cn.*
- *C. Jin and A. Zhou are with the School of Data Science and Engineering, East China Normal University, Shanghai 200062, China. E-mail: {cqjin, ayzhou}@sei.ecnu.edu.cn.*

Manuscript received 19 Feb. 2017; revised 10 July 2017; accepted 21 Aug. 2017. Date of publication 25 Aug. 2017; date of current version 3 Nov. 2017. (Corresponding author: Cheqing Jin.)
Recommended for acceptance by W. Zhang.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2017.2744619



(a) Speed-based outlier (b) Direction-based outlier

Fig. 1. Examples of trajectory outliers.

speed feature is small and the weighted linear sum of the distances for all the features is not large enough to make it different from most of the other trajectories.

Similarly, in hurricane landfall forecast scenario, as depicted in Fig. 1b, Tr_4 (in red) locates in very dense region, but the moving direction of it is opposite to its neighbors (move from east to west). Tr_4 is an outlier, indicating a hurricane whose wind direction changes suddenly. But in the traditional way, whether Tr_4 is an outlier or not depends on the weight setting of direction feature. Only when the weight of direction feature is large and the weighted linear sum of the distances for all the features is large enough to make it distinct from most of the others, Tr_4 would be identified as an outlier.

In order to address the aforementioned issue, we put forward a novel feature grouping-based mechanism, where all the features are divided into two groups, and the goal is to find outliers from a set of objects which are close to one another by the first group of features, but obviously different by the rest of features. According to the new scenario, Tr_2 and Tr_4 are regarded as outliers, because Tr_2 is close to its neighbors (Tr_1 and Tr_3) by spatial proximity but with different speed, Tr_4 is close to its neighbors by spatial proximity but different in the direction feature.

It is challenging to detect outliers upon trajectory data streams due to strict time—.and space—complexities. Aside from the aforementioned problem of feature selection, other key considerations could and should be taken into account when devising outlier detection method for trajectory stream. To deal with a stream of infinite positional point series, outlier detection technique must satisfy the requirements such as real-time response, lightweight execution, noise-tolerant and tracking the evolutionary property of outlier. Specifically, first, trajectory position information is transient. An outlier should be determined immediately once it occurs, which requires to handle the streaming data efficiently. Second, noise that is alike to outlier becomes the barrier of anomaly detection, and thus outlier detection technique should discern noise from true anomaly to improve detection precision. Finally, trajectory stream may evolve dynamically, and abnormal moving property of trajectory evolves gradually. Therefore, outlier detection technique should have ability to identify the evolutionary trajectory outlier as soon as possible.

Although there exist considerable researches on trajectory outlier detection, relatively few researches have worked on the streaming scenarios [16], [17]. Bu et al. [16] presented an abnormal trajectory segment detection technique according to the dissimilar property between one trajectory segment and most segments adjacent to it in time. However, the local continuity characteristic presented in [16] does not fit for abnormal moving object detection in massive-scale trajectory stream populated with numerous objects. Yu et al. [17] proposed a Minimal EXamination (MEX) framework to detect neighbor-based moving object outlier based on the neighbour relationship among moving objects. Nevertheless, none of them considers feature grouping, so that the trajectory outliers in Fig. 1 cannot be detected effectively.

In this paper, we introduce a novel framework to identify local anomaly trajectory fragment and evolutionary anomaly moving object in streaming scenarios. It is comprised of two components, including trajectory simplification phase, and outlier detection phase. During the simplification phase, appropriate fragments of raw trajectories at each timebin (including m timestamps, $m \geq 1$) are derived and the features of trajectory fragments are classified into two groups. During the detection phase, to estimate the degree of outlierness for trajectory fragments with regard to their neighboring fragments at each timebin, we first search local neighbors for each trajectory fragment by computing the similarity in the first group of features. Subsequently, we assign a local anomaly factor for every trajectory fragment in terms of the difference in the second group of features between its neighbors and itself, and identify local anomaly trajectory fragments by comparing with a given local anomaly threshold. Finally, considering the evolving nature of streaming trajectories, the local anomaly factor of moving object's trajectory fragment at each timebin is accumulated with historical local anomaly factor of its older trajectory fragments multiplied by a decay function, to generate its evolving anomaly factor. When the evolving anomaly factor of a moving object exceeds a given evolutionary anomaly threshold, it is regarded as a moving object outlier. Specifically, the contributions of this paper are summarized below.

- We propose a feature grouping-based mechanism to detect outliers upon trajectory streams, where the outlier is a trajectory fragment or an object that is close to its neighbors by some features, but significantly different from vicinities by the rest of features. To the best of our knowledge, there exists no prior work on this mechanism.
- We present two trajectory outlier definitions, *TF-outlier* and *MO-outlier*, to characterize local trajectory fragment outlier and evolutionary moving object outlier respectively.
- We propose a two-phase trajectory outlier detection framework, and then present two outlier detection approaches upon trajectory stream (denoted as *TODS* and *OTODS* respectively), to identify both types of trajectory outliers.
- We conduct a comprehensive series of experiments on three real data sets to manifest the efficiency and the effectiveness of our proposal, as well as the superiority to other congeneric approach.

The remainder of this paper is structured as follows. Section 2 reviews the related work in literature. In Section 3, the preliminary concepts are introduced and the problem is defined formally. In Section 4, we outline the scheme and elaborate the details of *TODS* and *OTODS* algorithms. In Section 5, a series of experiments are conducted on three data sets to evaluate our proposal. Finally, in the last section, we succinctly conclude the whole article and point out future directions.

2 Related Work

Trajectory data analysis enables us to understand the moving behaviors of moving objects [18], [19], [20], [21]. Trajectory outlier detection is one of the most valuable analysis tasks. In different scenarios, trajectory outlier has been characterized as diverse notions, such as abnormal sub-trajectory [10], abnormal trajectory [22], [23], [24], abnormal moving object [17], abnormal road segment [25], [26], [27], [28], and abnormal event [28], etc. Existing detection techniques [29] involve classification-based approach [30], [31], historical similarity based approach [9], [11], [12], [25], [26], [27], [28], distance-based approach [10], [16], [17], [22], [23], [24], [32], direction—and density-based approach[33], isolation-based approach [34], [35], [36], etc.

Anomaly Detection in Static Trajectory Data Set. The most intuitive trajectory outlier detection technique is to build a classification model based on a labeled data set to differentiate outlier from normal data [30], [31]. Nevertheless, it is difficult to obtain a good training data set for training and validating the anomaly detection model. Since the classification-based approach requires high computational stages to train and rebuild the classifier periodically, such approach is not tailored to detect outliers upon streaming trajectories. The historical similarity based approach attempts to construct a global feature model to detect outliers by mining frequent patterns on historical trajectories. Such a global feature model can attain higher detection precision if ignoring the evolutionary property in streaming context [9], [11], [12], [25], [26], [27], [28], [33]. Nevertheless, the detection models built upon historical trajectories are still unable to identify new abnormal behaviors in trajectory streams. With no need of any prior distributional assumptions, the distance-based outlier detection approaches define outliers as the trajectories far from most of other trajectories. Knorr et al. regarded object's whole trajectory as the basic unit and attempted to find abnormal trajectory that the majority of the trajectories stand apart from it [22], [23], [24]. In order to detect the abnormal sub-trajectory, Lee et al. proposed a partition-and-detect framework (*TRAOD* method) that integrates the distance-based and density-based approaches [10]. Before detecting trajectory outlier, they partitioned the trajectories into line segments using an approximate method based on minimum description length (MDL) principle [15]. Due to $O(n)$ time complexity of this approximate method, we adapt it to simplify trajectory stream data, as illustrated in Section 4.1. But *TRAOD* algorithm still focuses on discovering a few objects that are distant from most of the other objects. Additionally, the execution overhead of distance-based approach is at least quadratic with respect to the number of trajectories, so that it is unsuitable for streaming trajectories.

Anomaly Detection in Trajectory Stream. The aforementioned approaches require to scan the data set multiple times, whereas the streaming outlier detection algorithms tend to run in one-pass manner. Due to huge volume, rapid updating, sparsity and skewed distribution of streaming trajectories, there exist relatively few researches on outlier detection upon trajectory streams [16], [17], [35]. By comparing the difference between historical trajectories and an ongoing trajectory, Chen et al. presented *iBOAT* algorithm to detect anomalous sub-trajectory in real time [35]. Bu et al. proposed a local clustering-based approach for identifying an outlying trajectory segment of single moving object [16]. To reduce the computation and memory costs in outlier detection, [16] utilizes local cluster, a piecewise VP-tree based index structure and a minimum heap as pruning mechanism. However, this approach is not well suited for the trajectory stream, which is populated with voluminous objects and the moving patterns get more complex and dynamic. Aiming at discovering the abnormal moving objects over high-volume trajectory streams, Yu et al. presented neighbor-based trajectory outlier definitions (point neighbor based outlier and trajectory neighbor based outlier) [17], which consider the spatial similarity among objects and the duration of spatial similarity over time. Then they proposed a MEX framework equipped with three fundamental optimization principles (minimal support examination, time-aware examination, and lifetime-triggered detection) for detecting both kinds of outliers.

Nevertheless, the above approaches determine the outlierness based solely on spatial proximity among trajectories. They fail to consider the difference of moving behavior property among trajectories with respect to their local neighborhood. While such behavior difference indicates semantic spatial relationship, and it can be more valuable than spatial proximity in determining the cause of the outlierness. Moreover, the aforementioned methods scarcely consider the evolutionary nature of trajectory outlier. Therefore, they cannot be directly applied to solve our proposed outlier detection problem in Section 1. In order to detect local outliers and evolving outliers with respect to their neighbors without influence of noise disturbance, a feature grouping-based outlier detection technique is required to capture the behavior difference (or outlierness) among each trajectory and its local neighbors in real-time, and obtain the evolving outlierness of each trajectory by accumulating its historical outlierness and instant outlierness.

TABLE 1
List of Notations

Notation	Definition
S	the trajectory stream
T_c	the current timebin in trajectory stream
N	the window size
p_i	the location of an object at the timestamp t_i
tf	a trajectory fragment
TF_c	the set of trajectory fragments at current timebin
M	the number of features per trajectory
d	the distance threshold
d_a	the ally threshold
ρ	the local outlier threshold
ι	the evolutionary anomaly threshold

3 Problem Definition

In this section, we introduce some preliminary concepts, and formalize the problem of outlier detection upon trajectory streams. Table 1 summarizes major notations used in the rest of this paper.

Definition 1 (Trajectory Stream). *A trajectory stream refers to sequences of positional records of multiple moving objects, which is denoted as* $S = \{p_1^{(1)}, p_1^{(2)}, \ldots, p_2^{(1)}, p_2^{(2)}, \ldots\}$, *where* $p_i^{(j)}$ *is the location of an object* $o^{(j)}$ *at timestamp* t_i *in 2-D space, i.e.,* $p_i^{(j)} = (x_i^{(j)}, y_i^{(j)})$.

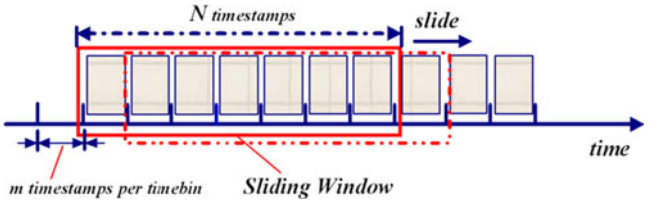


Fig. 2. Example of a sliding window with size of N timestamps.

Due to different sampling rates of various moving objects, to guarantee each moving object report its location at least once in a time interval, we utilize the term "timebin" (denoted as T) to describe a basic time interval. One timebin includes m timestamps ($m \geq 1$). Correspondingly, a trajectory stream is regarded as an infinite sequence of trajectory points ordered by timebins. In order to limit the infinite trajectory stream to a specified finite set of records within a given time horizon, we employ the concept of time-based sliding window. It can be the stream elements that arrived within the most recent timebins, e.g., an hour. It is commonly used for discounting obsolete data as new ones come in, which keeps the window fresh. Let N represent the window size. Whenever the window slides forward, it moves forward by 1 timebin, as shown in Fig. 2.

Definition 2 (Trajectory). *The trajectory of an object o, denoted as $Tr_o = \{(p_1, t_1), (p_2, t_2), \ldots (p_n, t_n)\}$, is a sub-sequence of S affiliated to o. Such records arrive in chronological order, where p_i is the location of o at timestamp $t_i, \forall i < j, t_i < t_j$.*

It is space-efficient to summarize each trajectory by reserving a small number of samples. For example, $\{p_1, p_2, \ldots, p_{100}\}$ can be summarized by $\{p_1, p_{11}, p_{21}, \ldots p_{91}\}$. How to select appropriate samples is challenging. We provide a trajectory simplification method in this paper (see Section 4.1 later), where each trajectory is simplified by a set of characteristic points, and every two consecutive characteristic points are connected into a trajectory fragment.

Definition 3 (Trajectory Fragment). *The fragment of a trajectory Tr_o, denoted as $tf = \{(p_i, t_i), (p_j, t_j)\}$ $(i < j)$, is a line segment connecting two consecutive characteristic points (p_i, p_j).*

Henceforth, we use $TF_c(tf_1, tf_2, \ldots)$ to represent the set of trajectory fragments derived by simplifying the trajectory stream at current timebin. Let $F = f_1, \ldots, f_M$ denote M features of a trajectory fragment. We divide all features into two groups. The first group $(f_1, \ldots, f_b)$, called as *Similarity Feature* (or, simply, *SF*), is used to find the spatial neighbors of each trajectory fragment, and the second group $(f_{b+1}, \ldots, f_M)$, called as *Difference Feature* (or, simply, *DF*), is used to identify the trajectory fragment which is obviously distinct from its spatial neighbors.

For instance, in hurricane landfall forecast application, track information of hurricane involves the hurricane's position in latitude and longitude, maximum surface wind speed, and minimum central pressure within a certain period. To discover the abnormal behaviors of hurricanes that are significantly different from their neighboring trajectories with respect to wind speed and central pressure, we regard latitude and longitude as *SF*, and choose maximum surface wind speed and minimum central pressure as *DF*.

Let $w_1, \ldots, w_M$ denote the weight of each feature respectively, $\sum_{l=1}^{M} w_l = 1$, and $dis_l(tf_i, tf_j)$ denote the distance between two trajectory fragments (tf_i and tf_j) by feature f_l. Note that $dis_l(tf_i, tf_j)$ can be any typical distance metric, such as Euclidean distance, DTW distance and Hausdorff distance, etc. Furthermore, $Diff_1$ and $Diff_2$ denote the distance between tf_i and tf_j by SF or DF, respectively.

$$Diff_1(tf_i, tf_j) = \sum_{l=1}^{b} w_l \cdot dis_l(tf_i, tf_j) \tag{1}$$

$$Diff_2(tf_i, tf_j) = \sum_{l=b+1}^{M} w_l \cdot dis_l(tf_i, tf_j) \tag{2}$$

To identify anomaly trajectory fragment at each timebin, we search neighbors for each trajectory fragment in terms of SF, and then estimate the anomaly degree of each trajectory fragment within its similar neighborhood according to DF.

Definition 4 (Neighborhood of Trajectory Fragment). *Given a threshold d, the neighborhood of trajectory fragment tf_i at timebin T_c contains all the fragments whose distances from tf_i are not larger than d by SF, i.e.,*

$$N_{T_c}(tf_i) = \{tf_j \in TF_c | Diff_1(tf_i, tf_j) \leq d\}$$

We adopt the idea of Local Outlier Factor (LOF) [37] to measure the outlierness of trajectory fragment. LOF, a density-based outlier detection technique by considering the neighborhood density of each element relative to that of its nearest neighbors, does not need to learn data distribution in advance, and is capable of detecting outliers with respect to the density of their local neighbors. Based on the idea of LOF, we introduce the concept of the local difference density to measure the difference in DF between each trajectory fragment and its adjacent trajectory fragments, and local anomaly factor (LAF) to estimate the anomaly degree of a trajectory fragment.

Definition 5 (Local Difference Density). *The local difference density of tf_i at timebin T_c is defined as*

$$ldd_{T_c}(tf_i) = \frac{|N_{T_c}(tf_i)|}{\sum_{tf_j \in N_{T_c}(tf_i)} Diff_2(tf_i, tf_j)}.$$

Definition 6 (Local Anomaly Factor). *The local anomaly factor of a trajectory fragment tf_i at timebin T_c is defined as*

$$LAF_{T_c}(tf_i) = \frac{\sum_{tf_j \in N_{T_c}(tf_i)} \frac{ldd_{T_c}(tf_j)}{ldd_{T_c}(tf_i)}}{|N_{T_c}(tf_i)|}.$$

In general, a larger LAF indicates that a trajectory fragment is more likely to be an outlier candidate. Therefore, a local anomaly trajectory fragment is a trajectory fragment with LAF above a given threshold, see the definition below.

Definition 7 (Local Anomaly Trajectory Fragment). *Given a local outlier threshold ρ, trajectory fragment tf_i at timebin T_c is called a local anomaly trajectory fragment, or in short,* TF-outlier, *iff $LAF_{T_c}(tf_i) > \rho$.*

TF-outlier is a fragment with DF significantly different from its neighborhood at current timebin. For instance,

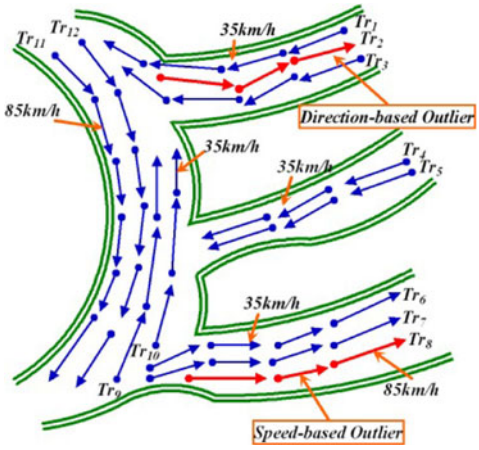


Fig. 3. Illustration of special cases.

reconsidering that trajectory (tr_2) with a velocity of 7 km/h in Fig. 1, it is obviously a local outlier with regard to its neighbors in other roads with average velocity of 44 km/h.

However, due to noise disturbance, a trajectory fragment with a larger LAF may be false *TF-outlier* if not given an appropriate local anomaly threshold. We take into account evolving nature of trajectory stream in solving this problem. The moving behaviors of objects keep evolving in streaming scenarios. An object which has normal trajectory fragment at current timebin may evolve into an outlier in the future. Distinct from anomaly detection of trajectory fragments at each timebin, we shall continuously capture the evolving trajectory outlier from the beginning of the trajectory stream via an evolving outlierness measurement. Notice that this evolving outlierness measurement should not only consider the local anomaly factor of the object's trajectory fragment at current timebin, but also take into account the influence of historical outlierness of object's older trajectory fragment.

That the sliding window model completely eliminates the historical data is undesirable for this issue. An alternative approach is to employ an idea of time decay to reduce the importance of historical outlierness. Specifically, for each moving object at per timebin, we accumulate the products of all the local anomaly factor of its historical trajectory fragments and a forward time decay function [38], and add the local anomaly factor of its trajectory fragment at current timebin, to obtain its evolutionary anomaly factor. The forward decay is computed on the amount of time between the arrival timebin of each historical trajectory fragment of object and a landmark, where the arrival timebin of the oldest trajectory fragment of each object, denoted as T_s, is used as the landmark for each object.

Definition 8 (Evolutionary Anomaly Factor). *Given the oldest timebin T_s, the current timebin T_c ($T_s \leq T_c$), and a monotone non-decreasing function g, the evolutionary anomaly factor of a moving object o is defined as*

$$EAF_{T_c}(o) = \sum_{T_k=T_s}^{T_c} \frac{g(T_k - T_s) LAF_{T_k}(tf_i)}{g(T_c - T_s)}.$$

Definition 9 (Evolutionary Anomaly Moving Object). *Given an evolutionary anomaly threshold ι, a moving object o at timebin T_c is an outlier, or in short,* (MO-outlier*), iff* $EAF_{T_c}(o) > \iota$.

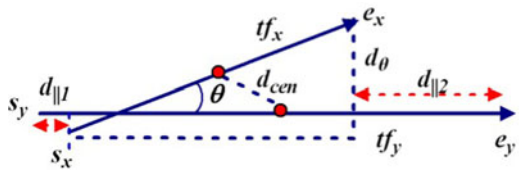


Fig. 4. Difference measurement of direction-based outlier.

Finally, we summarize the problem definition below.

Problem Statement. Given local outlier threshold ρ and evolutionary anomaly threshold ι, our goal is to continuously discover all the local anomaly trajectory fragments (*TF-outlier*) and the evolutionary anomaly moving objects (*MO-outlier*) upon trajectory streams.

3.1 Special Cases

3.1.1 Driving in Reverse

A vehicle that drives in reverse reflects in its trajectory whose moving direction is opposite to its vicinities. As exemplified in Fig. 3, the direction of trajectory Tr_2 is opposite to its neighbors (Tr_1 and Tr_3) at current timebin. To find the behavior of driving in reverse, we choose latitude and longitude as SF, and direction (calculated by latitude and longitude) as DF. Based on SF, we first lookup for the neighbors of each trajectory fragment using $Diff_1$, which is the weighted sum of the center point distance (denoted as d_{cen}) and the parallel distance (denoted as $d_\|$). For any two trajectory fragments tf_x and tf_y shown in Fig. 4, tf_y is longer than tf_x, without loss of generality, where $d_{cen} = \| cen_x - cen_y \|$; $d_\| = \frac{d_{\|1}+d_{\|2}}{2}$. Then we use $Diff_2$ (Euclidean distance) to measure the local difference density as well as the local anomaly degree of each trajectory fragment on DF between its neighbors and it. Here, $Diff_2$ refers to the angle distance (denoted as d_θ) computed on DF. Let $\| tf_x \|$ denote the length of tf_x, we assign θ the smaller intersecting angle between tf_x and tf_y, then d_θ is defined as follows.

$$d_\theta = \begin{cases} \| tf_x \| \times sin(\theta), & 0^\circ \leq \theta < 90^\circ \\ \| tf_x \|, & 90^\circ \leq \theta < 180^\circ \end{cases}$$

Finally, we identify *TF-outlier* Tr_2 at current timebin, as illustrated in Fig. 3, which indicates a car is driving along the wrong way.

3.1.2 Overspeeding

Overspeeding is a severe road traffic violation action, and easily leads to the occurrence of heavy traffic accidents. As illustrated in Fig. 3, the overspeeding behavior of vehicle can be depicted as a trajectory Tr_8 with high speed (85 km/h), whereas its neighbors (Tr_6 and Tr_7) keep slow speed (35 km/h). To find the overspeeding behavior, we choose latitude, longitude and direction (calculated by latitude and longitude) as SF, and then the speed as DF. To begin, we find neighboring trajectories with the same direction by using $Diff_1$ on SF, which is the weighted sum of d_{cen}, $d_\|$ and d_θ (in Fig. 4). Then we use $Diff_2$ (Euclidean distance) to measure the local difference density and the local anomaly degree of each trajectory fragment on DF between its neighbors and it, based on which *TF-outlier* (Tr_8) is

identified. Over the next several timebins, we proceed to measure the local anomaly degree and the evolutionary anomaly factor of trajectory fragments. When the evolutionary anomaly factor of the moving object that Tr_8 belongs to exceeds ι, we detect it as the *MO-outlier*, which indicates that it is a speeding car.

4 Framework: Local and Evolutionary Outlier Detection in Trajectory Stream

In this section, we propose a framework to continuously identify local anomaly trajectory fragments and evolutionary anomaly moving objects upon streaming trajectories. In our framework, each incoming trajectory will be split into a set of consecutive trajectory fragments. As illustrated in Algorithm 1, our framework is comprised of two phases, including trajectory simplification phase (at lines 1.1-1.4) and outlier detection phase (at lines 1.5-1.7). During the first phase, appropriate fragments of raw trajectories at each timebin are derived, the pseudocode of trajectory simplification function is presented in Algorithm 2. During the second phase, at each timebin, local anomaly factor and evolutionary anomaly factor of each trajectory fragment are computed, and based on which local anomaly trajectory fragments as well as evolutionary anomaly moving objects are detected.

Algorithm 1. Local and Evolutionary Outlier Detection upon Trajectory Stream

Input: S: a trajectory stream
Output: (1)A_{st}: a set of local anomaly trajectory fragments;
(2)A_o: a set of evolutionary anomaly moving objects;
1: **foreach** trajectory Tr at incoming timebin T_c of S **do**
2: $Tr_{simp} \leftarrow TraSimp(Tr)$;
3: $Tr_{all} \leftarrow Tr_{all} \cup Tr_{simp}$;
4: Generate sets of trajectory fragments from Tr_{all};
5: **foreach** *set of trajectory fragments* TF_c *at incoming timebin* T_c of S **do**
6: $(A_{st}, A_o) \leftarrow TODS(TF_c)$ (or $(A_{st}, A_o) \leftarrow OTODS(TF_c)$);
7: **return** A_{st} and A_o;

4.1 Trajectory Simplification

To accelerate trajectory outlier detection, each trajectory is simplified into a small number of characteristic points with least information loss. Specifically, when the behavior of one positional point is quite different from its previous points, this point is determined as a characteristic point. For a trajectory $Tr = \{p_1, p_2, \dots p_n\}$, it can be simplified with k characteristic points, namely, $Tr_{simp} = \{p_{l1}, p_{l2}, \dots p_{lk}\}$ ($1 \leq l_1 < l_2 < \cdots < l_k \leq n$). Each two adjacent characteristics points are connected to form a trajectory fragment (denoted as $tf(p_{li}, p_{li+1}), 1 \leq i < k$). Fig. 5 illustrates an example trajectory $Tr = \{p_1, p_2, \dots p_{10}\}$ and its characteristic points (highlighted in red) after simplification, i.e., $Tr_{simp} = \{p_1, p_3, p_5, p_9, p_{10}\}$. An appropriate simplification method must ensure that the number of characteristic points is as little as possible (conciseness), and the difference between the original trajectory and its simplified edition is as small as possible (preciseness). However, conciseness and preciseness are conflicting properties. This needs to seek an optimal trade-off between them.

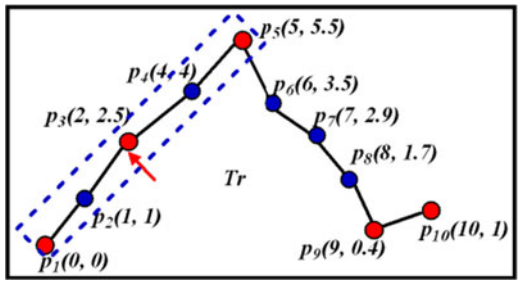


Fig. 5. A trajectory and its characteristic points.

Algorithm 2. TraSimp (Trajectory Simplification)

Input: Tr: a trajectory $\{p_1, p_2, \dots p_n\}$ in trajectory stream;
Output: Tr_{simp}: a set of all extracted characteristic points;
1: $Tr_{simp} \leftarrow \{p_1\}, s \leftarrow 1$;
2: /*s denotes the starting index and c denotes the current index*/;
3: $len \leftarrow 1, D_{max} \leftarrow 0, I_{max} \leftarrow s$;
4: /*len denotes the length of trajectory fragment*/;
5: **while** $(s + len \leq n)$ **do**
6: $c \leftarrow s + len$;
7: $DIFF \leftarrow MDL_{np}(p_s, p_c) - MDL_p(p_s, p_c)$;
8: **if** $DIFF > D_{max}$ **then**
9: $D_{max} \leftarrow DIFF$;
10: $I_{max} \leftarrow c$;
11: **if** $DIFF < 0$ **then**
12: $Tr_{simp} \leftarrow Tr_{simp} \cup \{p_{I_{max}}\}$;
13: $s \leftarrow I_{max} + 1$;
14: $len \leftarrow 1, D_{max} \leftarrow 0, I_{max} \leftarrow s$;
15: **else**
16: $len \leftarrow len + 1$;
17: $Tr_{simp} \leftarrow Tr_{simp} \cup \{p_n\}$;
18: **return** Tr_{simp};

Trajectory simplification (or segmentation) techniques include temporal interval based method [39], distance based method [40], representativeness based method [41] and trajectory shape based method [15]. In our trajectory simplification phase, we employ MDL principle and adapt the approximate method in [15]. The cost of MDL is represented as the sum of $L(H)$ and $L(D|H)$ (both encoded in bits), where $L(H)$ (conciseness measurement) denotes the sum of the length of all trajectory fragments, and $L(D|H)$(preciseness measurement) represents the sum of the differences between a trajectory and its derived trajectory fragments. We utilize $Diff_1$ by SF (i.e., coordinate information of positional point) to measure the difference, formally, we have $L(H) = \sum_{j=1}^{m-1} \log_2(Diff_1(p_{l_j} p_{l_{j+1}}))$, $L(D|H) = \sum_{j=1}^{m-1} \sum_{k=l_j}^{l_{j+1}-1} \log_2(Diff_1(p_{l_j} p_{l_{j+1}}, p_k p_{k+1}))$ An optimal simplification is achieved by minimizing the cost of MDL. In [15], an approximate solution is to achieve local optima on MDL cost of each trajectory fragment. Let $MDL_{np}(p_i, p_j)$ denote the MDL cost when there is no simplification between p_i and p_j, and $MDL_p(p_i, p_j)$ denote the MDL cost when treating p_i and p_j as two characteristic points, then $MDL_p(p_i, p_j) = L(H) + L(D|H)$ and $MDL_{np}(p_i, p_j) = L(H)$. Intuitively, the goal of local optimum is to find a long trajectory fragment (p_i, p_j). It ensures that no point

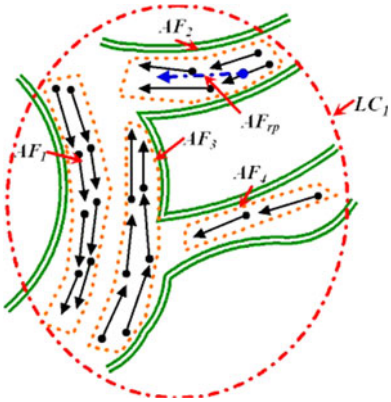


Fig. 6. One *Local Micro-cluster* and its *Ally Fragments*.

(e.g., p_k) between p_i and p_j satisfies $MDL_p(p_i, p_k) \geq MDL_{np}(p_i, p_k)$. If $MDL_p(p_i, p_{j+1}) \geq MDL_{np}(p_i, p_{j+1})$, point p_j is chosen as a characteristic point. Though this characteristic selection method guarantees good time complexity, it cannot ensure p_j is the point to minimize the cost of MDL.

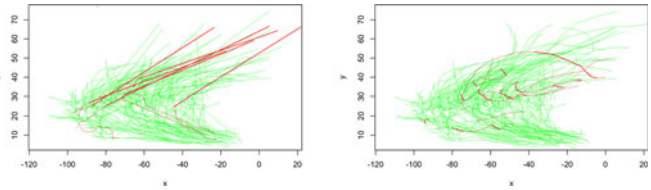
In view of this, as shown in Algorithm 2, when we calculate MDL_p and MDL_{np} for each point in a trajectory, we record two values: the maximal difference (denoted as D_{max}) between MDL_p and MDL_{np}, and the index of that point (denoted as I_{max}) with maximal difference in the traversed points so far (at lines 2.5-2.10). Each time a point is traversed, we compare the difference (denoted as $DIFF$) between its MDL_p and MDL_{np} with D_{max}, and set D_{max} to whichever is larger, and then update the value of I_{max}. When $DIFF < 0$, we choose $p_{I_{max}}$ rather than the immediately previous point as a characteristic point, and lookup for the next characteristic point start with $p_{I_{max}+1}$ (at lines 2.11-2.14).

Example 4.1. Let's consider a trajectory Tr of 10 points in Fig. 5. When we traverse point p_5 in Tr, because $MDL_p(p_1, p_5) \geq MDL_{np}(p_1, p_5)$, p_4 (the previous point of p_5) is chosen as a characteristic point by [15]. Distinct from that, because the difference between $MDL_{np}(p_1, p_4)$ and $MDL_p(p_1, p_4)$ is 1.17, the difference between $MDL_{np}(p_1, p_2)$ and $MDL_p(p_1, p_2)$ is 0, the difference between $MDL_{np}(p_1, p_3)$ and $MDL_p(p_1, p_3)$ is 3.03, $D_{max} = 3.03$, $I_{max} = 3$. So we choose p_3 instead of p_4 as a characteristic point. In this way, we simplify the rest points of Tr. This characteristic point selecting method can achieve a local optima better, which is demonstrated in Section 5 (as shown in Fig. 7).

4.2 TF-Outlier and MO-Outlier Detection

After the trajectory fragments are obtained by trajectory simplification at per timebin, we need to identify *TF-outlier* and *MO-outlier* with respect to their local neighborhood. The detailed description of *TODS* (short for TF-outlier and MO-outlier Detection upon Trajectory Stream) algorithm is presented in Algorithm 3.

At each timebin, the neighborhood of each trajectory fragment is determined by a distance threshold d (Definition 4). However, a fixed value of d is not well suited for searching neighbors in the regions of different density. The intuition is that the trajectory fragments are closer to each other in dense region, but depart from each other in sparse region. Therefore, setting a smaller distance threshold



(a) $TRAOD$ (b) $TODS$

Fig. 7. Outlier detection result for $BestTrack$.

empower the trajectory fragments have no neighbors in sparse regions. To solve this problem, we take density into account in distance threshold setting. The density of a trajectory fragment at T_c is determined by the amount of trajectory fragments within the distance from it, and such a distance can be calculated by the standard deviation (denoted as σ) of pairwise distances between trajectory fragments arrived at T_c, i.e., $Density_{T_c}(tf_i) = |N_{T_c}(tf_i)|$ in terms of distance σ. Accordingly, the value of d can be adjusted by multiplying with the ratio of average density of trajectory fragments arrived at T_c to its local density, namely, $d \cdot \frac{\frac{1}{|TF_c|}\sum_{tf_j \in TF_c} Density_{T_c}(tf_j)}{Density_{T_c}(tf_i)}$. Through adaptive adjustment of d, the value of d rises in the sparse region, and decrements in a dense region. In this way, we guarantee that the trajectory fragments have neighbors in sparse regions and they will not miss outlier detection.

Algorithm 3. *TODS* (*TF-outlier* and *MO-outlier* Detection upon Trajectory Stream)

Input: TF_c: a set of trajectory fragments at timebin T_c;
Output: (1)A_{st}: a set of local anomaly trajectory fragments;
(2)A_o: a set of evolutionary anomaly moving objects;

1: **for** each trajectory fragment tf_i of each object o at TF_c **do**
2: find the local neighbors $N_{T_c}(tf_i)$ for tf_i;
3: $LAF_{T_c}(tf_i) \leftarrow \frac{\sum_{tf_j \in N_{T_c}(tf_i)} \frac{ldd_{T_c}(tf_j)}{ldd_{T_c}(tf_i)}}{|N_{T_c}(tf_i)|}$;
4: **if** $LAF_{T_c}(tf_i) > \rho$ **then**
5: $A_{st} \leftarrow A_{st} \cup \{tf_i\}$;
6: $EAF_{T_c}(o) \leftarrow EAF_{T_{c-1}}(o)\frac{g(T_{c-1}-T_s)}{g(T_c-T_s)} + LAF_{T_c}(tf_i)$;
7: **if** $EAF_{T_c}(o) > \iota$ **then**
8: $A_o \leftarrow A_o \cup \{o\}$;
9: **return** A_{st} and A_o;

Subsequently, to identify *TF-outlier*s at T_c, we compute local difference density and local anomaly factor for each trajectory fragment (at line 3.3). Through comparing the difference of DF between each trajectory fragment and its vicinity, we derive the local anomaly degree of each element at T_c. Generally, if a trajectory fragment has great difference from its neighbors on DF, the local anomaly factor attains a higher value. Trajectory fragments are identified as *TF-outlier*s when their local anomaly factors exceed a given local anomaly threshold ρ (at lines 3.4-3.5).

A trajectory fragment is identified as a *TF-outlier* at one timebin according to the difference of DF between it and its adjacent elements. But due to the disturbance of noise data, it is difficult to specify an appropriate local outlier threshold to identify *TF-outlier* accurately and reduce the false alarms. In addition, with the evolution of trajectories, the phenomenon that a moving object has anomaly trajectory fragments

in multiple timebins (maybe discontinuous) indicates the exceptional behaviors of this moving object. Thus, an effective detection approach is desirable to capture the evolving abnormal behaviors of moving objects. Through employing the idea of strengthening the influence of newly arrived trajectory data while lessening the influence of outdated data (may be noisy data), we derive the evolving anomaly factor of each moving object based on the local anomaly factors of its trajectory fragments at different timebins.

Since the evolving outlierness of a moving object has been more affected by the recent abnormal behaviors than historical abnormal behaviors, a decay function is utilized to calculate the evolutionary anomaly factor (Definition 8). To be specific, LAFs of trajectory fragments of each moving object at the previous timebins are multiplied by a decay function, and accumulated with LAF at current timebin to derive the evolutionary anomaly factor. Because backward decay mechanism require calculating the age of each item with respect to current timebin, which is varying over time. Additionally, every item must be revisited to compute its contribution for decayed LAF computation. As a result, we adopt the notion of forward time decay [38] based on measuring forward from a previous fixed point in time (called as $landmark$). To attain a slower decay on historical abnormality, we apply polynomial decay, specifically, $g(n) = n^2$. According to Definition 8, we have

$$\begin{aligned} EAF_{T_c}(o) &= \sum_{T_k=T_s}^{T_c} \frac{g(T_k - T_s) LAF_{T_k}(tf_i)}{g(T_c - T_s)} \\ &= \sum_{T_k=T_s}^{T_{c-1}} \frac{g(T_k - T_s) LAF_{T_k}(tf_i)}{g(T_c - T_s)} + LAF_{T_c}(tf_i) \\ &= EAF_{T_{c-1}}(o) \frac{g(T_{c-1} - T_s)}{g(T_c - T_s)} + LAF_{T_c}(tf_i) \end{aligned}$$

Hence, the value of $EAF_{T_c}(o)$ can be calculated by multiplying $EAF_{T_{c-1}}(o)$ with the ratio of $g(T_{c-1} - T_s)$ to $g(T_c - T_s)$, and adding it with $LAF_{T_c}(tf_i)$ (at line 3.6). When the evolving anomaly factor of the moving object o grows beyond threshold ι, o would be reported as an evolutionary anomaly moving object (at lines 3.7-3.8).

Time Complexity Analysis. For the trajectory fragments arrived at current timebin, the time complexity of $TODS$ algorithm is $O(n \log n)$ using *STR-tree* index technique, here, n is the maximum number of trajectory fragments arrived at current timebin. Therefore, after insertion of t timebins, the total time complexity of iterated $TODS$ algorithm is $O(tn \log n)$.

Actually, trajectory data arrive very rapidly, searching neighbors for each trajectory fragment at each timebin intrinsically becomes the computational bottleneck. Accordingly, whenever receiving new trajectory data at every timebin, both local outlier factor calculation and evolutionary anomaly factor calculation are required executing iteratively for each trajectory fragment. This involves comparatively high computational overhead especially when massive amount of trajectory data arrive at one timebin. To further improve efficiency of $TODS$ algorithm, a high-quality but significantly less costly technique is desirable to accelerate neighbor searching and reduce the amount of updates.

4.3 Optimized TF-Outlier and MO-Outlier Detection

Although trajectories are time-varying evolutionary in streaming scenario, it is observed that the neighbor relationship among trajectory fragments may be retained for one or more timebins. Motivated by this, we combine with the clustering technique, and exploit a new structure to store and maintain the neighbor relationship of trajectory fragments along the trajectory stream. To be specific, we characterize a micro-group of trajectory fragments within a specified proximity threshold, as $Ally\ Fragment$ (AF for short). Here, we utilize a representative trajectory fragment (denoted as AF_{rp}) to depict the overall characteristic of an $Ally\ Fragment$.

Definition 10 (Ally Fragment). *Given an ally proximity threshold d_a ($d_a < d$), an ally fragment AF is defined as a set of trajectory fragments such that: for any trajectory fragment $tf_i \in AF$, the distance between tf_i and AF_{rp} by SF is not more than d_a, i.e., $Diff_1(tf_i, AF_{rp}) \leq d_a$.*

The trajectory fragments in one ally fragment are tightly close to one another. The maintenance of ally fragment is triggered only when inserting newly arrived trajectory fragments or deleting the obsolete trajectory fragments. Based on the structure of ally fragment, we present an optimized $TODS$ algorithm, denoted as $OTODS$. The detailed algorithmic description of $OTODS$ is given in Algorithm 4. Initially, randomly selecting a trajectory fragment from newly arrived ones as seed, then merging it with its nearest trajectory fragments to generate one ally fragment in terms of ally threshold d_a (at lines 4.2-4.4). If its nearest ally fragment cannot be found, a new ally fragment will be created for it (at lines 4.7-4.8). This select-and-merge process iterates until all the arrived trajectory fragments at one timebin are absorbed into different ally fragments. For all the ally fragments, it is required to derive the representative trajectory fragment (computing method is similar to [21]). Also, with the progression of stream, it is critical to eliminate the influence of obsolete trajectory fragments from the existing ally fragments. It mainly involves the updates of representative trajectory fragments for influenced ally fragments with the insertion and deletion of trajectory fragments. That we regard ally fragment instead of trajectory fragment as basic unit when calculating ldd and LAF, can drastically reduce the amount of calculation.

To further reduce the cost of local anomaly factor calculation, our solution builds $Local\ Micro\text{-}cluster$ (LC for short) on ally fragments via hierarchical clustering (at line 4.9), instead of searching neighbors for each ally fragment. As illustrated in Fig. 6, four ally fragments (denoted as AF_1, AF_2, AF_3, and AF_4 respectively) are clustered into a $Local\ Micro\text{-}cluster$ (denoted as LC_1). The representative trajectory fragment of AF_2 is AF_{rp}. $Local\ Micro\text{-}cluster$ is distinct from temporal partitioning (defined as $Local\ cluster$) over trajectory stream in [16], it intuitively extends the neighbor relationship of ally fragments. According to Definition 10, the trajectory fragments in an ally fragment are tightly close to each other, and thus local anomaly factors of the trajectory fragments in an ally fragment are approximately the

same. Henceforth, evaluating local anomaly factor for each trajectory fragment is transformed into estimating local anomaly factor of ally fragment in local micro-cluster. The structure of ally fragment simply stores the neighbor relationship between objects rather than their spatial coordinates. The size of *Ally Fragment* is much smaller than that of *Local Micro-cluster*. Therefore, the structures of *Ally Fragment* and *Local Micro-cluster* can significantly accelerate the execution of *TODS* algorithm.

Essentially, insertion of new trajectory fragments or elimination of obsolete trajectory fragments only influences the updates of LAFs for the relevant elements, including (1) the ally fragments which newly inserted trajectory fragments belong to, (2) the ally fragments that outdated trajectory fragments deleted from, and (3) the other ally fragments in the local micro-clusters that newly inserted or deleted trajectory fragments belong to. We use Z_{INF} to denote the influenced set of ally fragments on insertion or deletion of trajectory fragments (at line 4.5, 4.8, 4.11). So the updates of LAFs after per insertion or elimination are limited on ally fragments in Z_{INF} and do not require recalculating LAFs for all of the ally fragments (at lines 4.14-4.15). In the following we proceed with describing the details of updating LAF when inserting or eliminating trajectory fragments as time goes by.

Algorithm 4. *OTODS* (Optimized *TF-outlier* and *MO-outlier* Detection upon Trajectory Stream)

Input: TF_c: a set of newly arrived trajectory fragments at timebin T_c;
Output: (1)A_{st}: a set of local anomaly trajectory fragments;
(2)A_o: a set of evolutionary anomaly moving objects;
1: Initialize the set of ally fragments Z;
2: **foreach** trajectory fragment tf_i of object o in TF_c **do**
3: **if** $\exists AF, Diff_1(tf_i, AF) \leq d_a$ **then**
4: Insert tf_i into AF;
5: $Z_{INF} \leftarrow Z_{INF} \cup \{AF\}$;
6: /*Z_{INF} denotes the influenced set of ally fragments on insertion or deletion of new trajectories*/;
7: **else**
8: Create AF_{new} for tf_i; Update Z and Z_{INF};
9: Assign the closest micro-cluster to AF_{new} using clustering method;
10: **foreach** outdated trajectory fragment tf_e **do**
11: Remove tf_e from the corresponding AF; Update Z and Z_{INF};
12: **foreach** ally fragment AF in Z_{INF} **do**
13: Find the influenced local micro-cluster LC_m for AF;
14: **foreach** ally fragment AF_j in LC_m **do**
15: $LAF_{T_c}(AF_j) \leftarrow \frac{\sum_{AF_y \in LC_m} \frac{ldd_{T_c}(AF_y)}{ldd_{T_c}(AF_j)}}{|LC_m|}$;
16: **if** $(LAF_{T_c}(AF_j) > \rho)$ **then**
17: **foreach** fragment tf_i in AF_j **do**
18: $A_{st} \leftarrow A_{st} \cup \{tf_i\}$;
19: **foreach** fragment tf_i in AF_j **do**
20: $EAF_{T_c}(o) \leftarrow EAF_{T_{c-1}}(o)\frac{g(T_{c-1}-T_s)}{g(T_c-T_s)} + LAF_{T_c}(tf_i)$;
21: **if** $EAF_{T_c}(o) > \iota$ **then**
22: $A_o \leftarrow A_o \cup \{o\}$;
23: **return** A_{st} and A_o;

Insertion of New Trajectory Fragments. The main work for inserting new trajectory fragments involves the updates of ldds, LAFs and EAFs for newly inserted ally fragments as well as affected existing ones. On the one hand, whenever new trajectory fragments are absorbed into existing or newly created ally fragments, ldds, LAFs and EAFs need to be calculated for the new records. On the other hand, with the insertion of new records into existing ally fragments, the other ally fragments in the same local micro-cluster that new records belong to, shall accordingly update their ldds. In addition, new records and affected records need to reassign the degree of outlierness for each element, i.e., LAF and EAF (at lines 4.14-4.22).

Elimination of Obsolete Trajectory Fragments. A certain amount of trajectory fragments will be eliminated due to their obsoleteness of each timebin, which even leads to some ally fragments need to be deleted. Similar to the insertion of new records, ldds, LAFs and EAFs of the affected trajectory fragments are required to be updated. Meanwhile, for all the local micro-clusters that deleted trajectory fragments belong to, ldds, LAFs and EAFs of the other ally fragments in the same local micro-cluster must be updated correspondingly.

Time Complexity Analysis. After insertion of t timebins, the time complexity of *OTODS* algorithm is $O(n \log n + (t-1)n_A \log n_A)$ using *STR-tree* index technique, where n is the maximum number of arrived trajectory fragments at a timebin, n_A is maximum number of ally fragments, and $n_A \ll n$. Thereby, the efficiency of *OTODS* algorithm is significantly improved as compared to *TODS* algorithm.

5 Experimental Study

In this section, we conduct extensive experiments to assess the effectiveness and efficiency of *TODS* and *OTODS*. Initially, to evaluate the effectiveness of *TODS* on the static trajectory data set, we utilize *TRAOD* [10] as the baseline approach to compare against *TODS* based upon the real data set. Further, we execute *TODS* and *OTODS* algorithms on two real data sets, to verify effectiveness of both algorithms over streaming trajectories. Finally, we compare the efficiency of *TODS* and *OTODS* algorithms on real data set, and then evaluate the influence of the key parameters on each algorithm.

TRAOD is the most effective distance-based trajectory outlier detection algorithm reported so far for static trajectory data set. It partitions each trajectory into a set of line segments (denoted as t-partition), and then detects outlying t-partitions and trajectory outliers. In detection phase, *TRAOD* determines the abnormity of a t-partition according to its insufficient amount of close trajectories relative to the whole trajectory data set. Note that a close trajectory defined in [10] not only needs to meet a given distance threshold, but also ensures that sufficient portions of it are close to a t-partition. Namely, the length of close portion of a close trajectory is not less than that of comparing t-partition. Additionally, *TRAOD* identifies the trajectory outliers according to the ratio of the sum of its outlying t-partitions' lengths to its total length. As a result, outlying t-partitions and trajectory outliers detected by *TRAOD* are probably very long. Distinct from *TRAOD*, our proposal aims to detect *TF-outlier* and *MO-outlier*. We initially find similar neighborhood for each trajectory fragment in terms of the specified

distance threshold rather than length limit of close trajectory. Then, we detect outliers within their local neighborhood. The rationale is that finding outliers locally is more accurate than doing so globally.

All codes, written in Java, are conducted on a PC with Intel Core CPU 3.6 GHz Intel i7 processor and 16 GB RAM. The operating system is Windows 10. Unless mentioned otherwise, the parameters are set below, the window size $N = 30$ minutes, $timebin = 2$ minutes.

5.1 Data Sets

We utilize three real trajectory data sets to evaluate our proposed methods, including atlantic hurricane track data set (hereafter termed $BestTrack$[1]), taxi operational data set of 2015 in Shanghai (hereafter termed $TaxiShanghai15$), and taxi trajectory data set of 2013 in Shanghai (hereafter termed $TaxiShanghai13$). The former data set is derived in Euclidean space, and the latter two on restricted road network.

Best Track records a wealth of hurricane's track information within the period between the year 1959 and 2010, containing features such as the hurricane's position in latitude and longitude, maximum surface wind speed, as well as minimum central pressure. We choose a portion (1990-2010) of the data set, which involves 221 trajectories and 6541 points.

Taxi Shanghai15 contains about 410 k trajectories derived by 13,600 taxis of Shanghai in a period from Apr.1 to Apr.30, 2015. Each GPS log is received at the rate of around once every minute. It has about 13,660 trajectories per day (about 114 million points) with six attributes of Vehicle ID, Time, Longitude and Latitude, Speed, and Taxi Status (free/occupied), etc.

Taxi Shanghai13 contains about 4,800 k trajectories generated by 13,400 taxis of Shanghai for three months, from October to December in 2013. We select a test area consists of 19 road segments (from Wuzhou Avenue, through Shenjiang road and Jufeng Road, to North Zhang Yang Road), and each road segment has one lane in each direction. It has about 53,356 trajectories per day (about 107 million points) with four attributes of timestamp, velocity, longitude and latitude coordinates. These positional records experience the periods of free-flow and congestion alternately throughout a day, and compose a real-time trajectory stream. The ground truth outlier set is manually verified through comparative velocity analysis for each trajectory fragment and its neighbors at each timebin by the volunteers. It involves a few trajectories with exceptional speed on some road segments, and the road segments that have obviously different speed from their neighbors. The labeling of *TF-outlier* (or *MO-outlier*) is determined by majority of volunteers' voting.

5.2 Effectiveness Evaluation

Results for Best Track. To verify the accuracy of $TODS$ on the static data set, we implement $TRAOD$ and $TODS$ on $BestTrack$ first. We fit $BestTrack$ in a time window, which enables $TODS$ to handle the static data set just like $TRAOD$. Fig. 7 visualizes the detection results of $TRAOD$ and $TODS$. We choose latitude and longitude as SF, regard direction (calculated by position information), minimum central pressure and wind speed as DF. To detect angular outlier, in the experiment of [10], more weights are put on the angular distance. Accordingly, we put more weights on direction feature and fewer weights on the other features in DF. Parameters of $TRAOD$ algorithm are set the same as [10], and the parameters of $TODS$ are empirically set as follows: $d = 40$, $\rho = 1.5$, and $\iota = 1.276$. As shown in Fig. 7, thin green lines represent normal trajectory partitions (or fragments), thick red lines represent anomaly trajectory t-partitions ($TRAOD$) or *TF-outlier*s ($TODS$), and thin red lines represent trajectory outliers ($TRAOD$) or *MO-outlier*s ($TODS$). Owing to the advantage of our improved simplified method, trajectory simplification result of $TODS$ is smoother than that of $TRAOD$, as illustrated in Fig. 7.

As shown in Fig. 7a, a total of 6 trajectory outliers with outlying t-partitions are detected by $TRAOD$. We can observe that most of outlying t-partitions and trajectory outliers are quite long and do not behave obviously different from their neighboring trajectories. As discussed earlier, the main reason is that close trajectories of a t-partition are determined by two issues [10], one is within the smallest distance, and the other is that the length of close trajectory must be larger than that of tested t-partition. If a t-partition L_i has not sufficient close trajectories with length greater than that of L_i, L_i is regarded as an outlying t-partition. Additionally, the trajectory outliers are identified according to the ratio of the sum of its outlying t-partitions' lengths to its total length. We observe that long t-partitions in the upper right region are identified as outlying t-partitions, because they are distant from most of other trajectories and few close t-partitions have a larger length than them. Correspondingly, since the sum of lengths of these t-partitions possesses a definite proportion relative to the trajectories that they belong to, such trajectories are identified as trajectory outliers even when the remainder portions of them are similar with their surrounding trajectories.

Whereas $TODS$ instead focuses on detecting *TF-outlier* and *MO-outlier* that are significantly distinct from their neighborhood according to DF. As illustrated in Fig. 7b, there are 12 *MO-outlier*s (their portions including *TF-outlier*s) detected by $TODS$ algorithm. Compared with trajectory outliers detected by $TRAOD$, *MO-outlier*s are not always long trajectories. But we can see that the directions of *MO-outlier*s are significantly different from their local neighborhood. Thus, the outlier detection result of $TODS$ is more reasonable than that of $TRAOD$.

Results for Taxi Shanghai15. We apply $OTODS$ algorithm on $TaxiShanghai15$ to detect traffic abnormal incidents in urban road network of Shanghai during the interval [8:00-8:30] on April 14 and 15, 2015 respectively. Longitude and latitude are chosen as SF, and speed is regarded as DF. We regard the real-time average speed of each $Ally\ Fragment$ as the velocity of corresponding road segment. We choose a test area (the road area surrounding the Middle Ring Road) to visualize the outlier detection result of $OTODS$ algorithm. As shown in Fig. 8, green lines represent the road segments where normal motion traces of taxis locate on, yellow lines represent *TF-outlier*s (the lanes in a certain road segments) within the interval [8:29-8:30], and red lines represent *MO-outlier*s (the lanes in a certain roads) within the interval [8:00-8:30]. We observe that the speeds of most outliers (*TF-outlier*s and *MO-outlier*s) are significantly higher than that of their

1. http://weather.unisys.com/hurricane/atlantic/

(a) 2015.4.14

(b) 2015.4.15

Fig. 8. Outlier detection result for $TaxiShanghai15$.

vicinities. High speed always implies traffic fluency on the road. During peak hours, the information that the roads with high speed is extremely useful and can be delivered to the public for optimal route planning in real time.

Furthermore, as seen from Fig. 8a and b, although there is no obvious difference between the traffic situations in the same time period on two separate days (Apr 14th and 15th), significant variations have emerged in the results of trajectory outlier detection, including the number of *TF-outlier*s and *MO-outlier*s, and the positions where outliers occur. In comparison with traffic situation on April 14 (Fig. 8a), the speeds of most of the roads (including the roads that outliers locate on and general roads) on April 15 (Fig. 8b) are obviously higher, especially for the roads nearby the Middle Ring Road. This case may be coincide with the execution of a known traffic administration rules published by the Shanghai Public Security Bureau. It says that some elevated highways (e.g., Middle Ring Road) extend the prohibited driving time by one hour from April 15, 2015, which includes morning rush hour [7:00-10:00] and evening rush hour [16:00-19:00]. This means that many vehicles are prohibited driving on elevated highways for more time. Therefore, owing to the reduction of vehicles, the velocities of two-way lanes in Middle Ring Road and small numbers of neighbor road segments

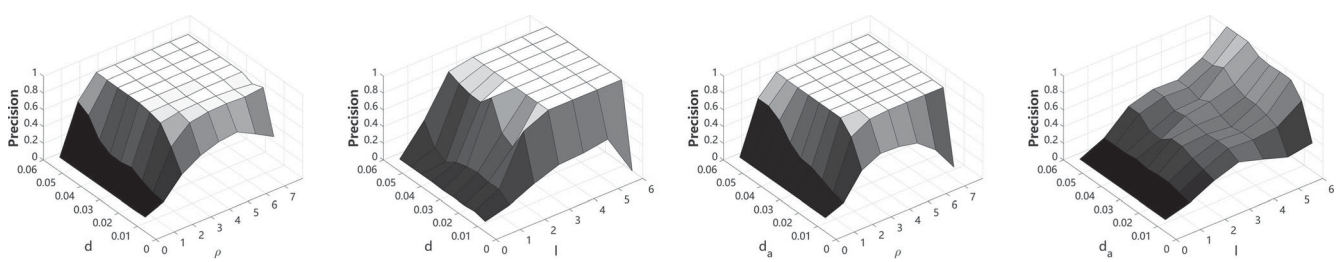
(a) Movement distribution (b) *TF-Outliers* detected by $TODS$

Fig. 9. Outlier detection result for $TaxiShanghai13$.

are accelerated within the interval [8:00-8:30], as illustrated in Fig. 8b. This is tally with new traffic control policy. It demonstrates that $OTODS$ method can be used to effectively identify traffic abnormal situation in real time.

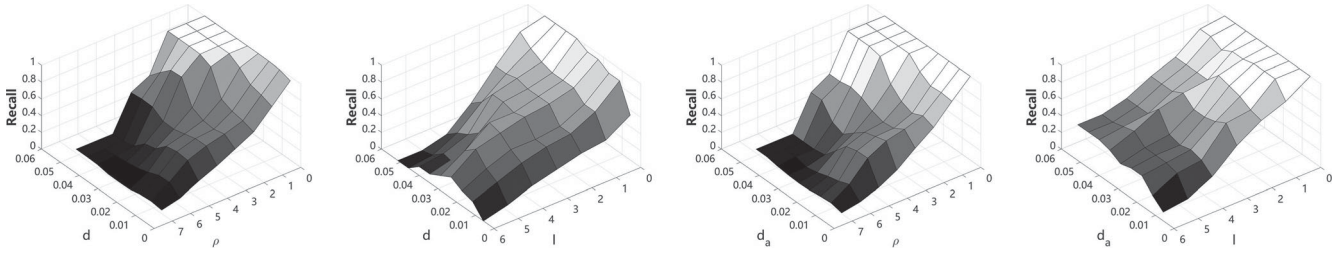
Results for Taxi Shanghai13. For effectiveness validation purpose, we also conduct $TODS$ and $OTODS$ algorithms on $TaxiShanghai13$. We aim to continuously identify *TF-outlier*s and *MO-outlier*s on $TaxiShanghai13$ according to velocity feature. Similarly, we choose longitude and latitude coordinate as SF, and regard velocity as DF. Take *TF-outlier*s detected by $TODS$ for example, the portions of test area and the outlier detection results (within [8:00-10:00] a.m. on October 8) are visualized in Fig. 9. Fig. 9a shows the movement distribution of taxis traces (in green). The average speed of most roads is not beyond $20\ \mathrm{km/h}$. The *TF-outlier*s (in red) detected by $TODS$ is illustrated in Fig. 9b. Thin red lines indicate that only a few abnormal trajectories (with exceptional speed of $50-60\ \mathrm{km/h}$) occur on parts of roads. *TF-outlier*s represented by thick red lines indicate a road segment where taxis travel with the significantly different speed from neighboring road segments. This outlier detection result is coincide with the ground truth outliers.

Metrics. In order to conduct a comparative analysis of the effectiveness for both algorithms, we use $Precision$, $Recall$ and $F-measure$ as the criteria measurement. They are defined as $Precision = \frac{|R \bigcap D|}{|D|}$, $Recall = \frac{|R \bigcap D|}{|R|}$, and $F-measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$. where R denotes the manually labeled outlier set, and D denotes the detected outlier set by our proposal. $Precision$ indicates how accurately the algorithm detects outliers and $Recall$ measures how completely outliers are detected. To deeply understand how the parameters impact outlier detection, we compare the results of both algorithms via $Precision$, $Recall$ and $F-measure$ by utilizing different thresholds (d, d_a, ρ and ι), as illustrated in Figs. 10, 11 and Fig. 12. Note that d (or d_a) uses the variation of latitude and longitude, the distance of 0.0001 is corresponding to about 11 meters.



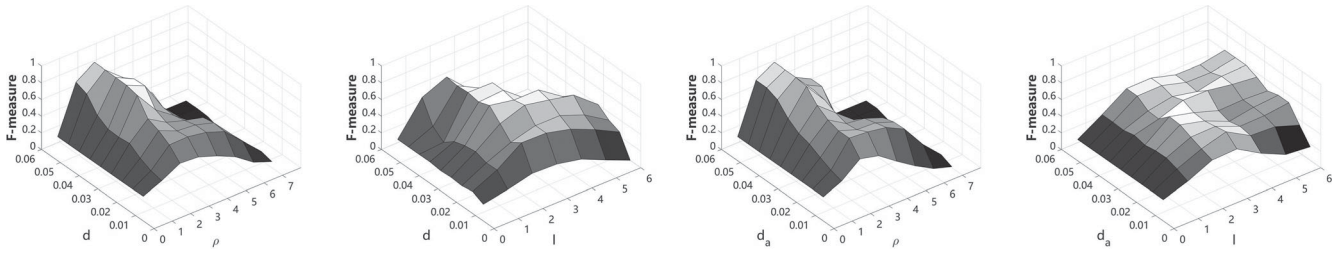
(a) *TF-outlier* of $TODS$ (b) *MO-outlier* of $TODS$ (c) *TF-outlier* of $OTODS$ (d) *MO-outlier* of $OTODS$

Fig. 10. $Precision$ ($TaxiShanghai13$).



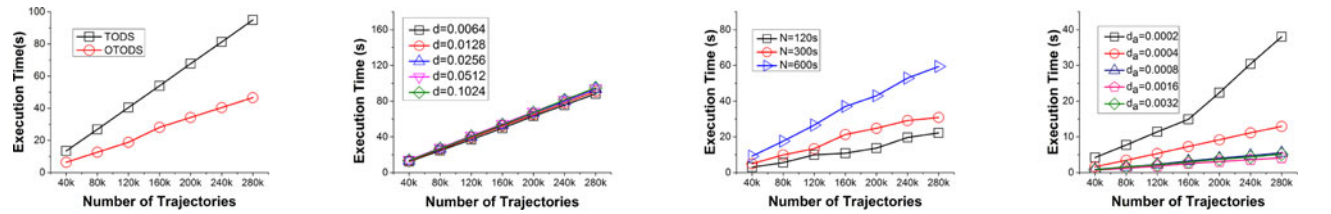
(a) *TF-outlier* of $TODS$ (b) *MO-outlier* of $TODS$ (c) *TF-outlier* of $OTODS$ (d) *MO-outlier* of $OTODS$

Fig. 11. $Recall$ ($TaxiShanghai13$).



(a) *TF-outlier* of $TODS$ (b) *MO-outlier* of $TODS$ (c) *TF-outlier* of $OTODS$ (d) *MO-outlier* of $OTODS$

Fig. 12. $F\text{-}measure$ ($TaxiShanghai13$).



(a) Execution time comparison between $TODS$ and $OTODS$ (b) Execution time of $TODS$ versus d (c) Execution time of $OTODS$ versus N (d) Execution time of $OTODS$ versus d_a

Fig. 13. Execution time comparison over $TaxiShanghai13$.

First, we investigate the precision of both algorithms, with the results under different distance threshold (d or d_a) and anomaly threshold (ρ or ι) plotted in Fig. 10. From Figs. 10a and c, for *TF-outlier*, $TODS$ and $OTODS$ algorithms have high precision when $\rho \geq 2$, because local anomaly factors of most *TF-outlier*s reach about 2. While $OTODS$ uses a smaller d_a ($d_a = 0.0032$) to attain the same $Precision$ as $TODS$ with a larger d ($d = 0.0256$), this is so because a smaller d_a enables all the trajectory fragments in one ally fragment closer to one another. Accordingly, LAF of such ally fragment represents the outlierness of trajectory fragments in it more precisely. In addition, comparing *MO-outlier* detected by $TODS$ (Fig. 10b) with that by $OTODS$ (Fig. 10d), $TODS$ has a higher precision than $OTODS$, and $OTODS$ obtains nearly the same precision as $TODS$ when $\iota = 6$. This is in line with our intuition that anomaly evaluation of trajectory fragments is transformed into that of ally fragments, and the average outlierness of ally fragments guarantees approximately the same high precision as originally trajectory fragments.

Second, we report the recall rate of both algorithms under different distance threshold (d or d_a) and different anomaly threshold (ρ or ι), as shown in Fig. 11. It is observed that $TODS$ and $OTODS$ algorithms have almost the same recall rate. Meanwhile, the recall rate of both algorithms gradually drops as anomaly threshold (ρ or ι) increases, and even further increasing the value of d or d_a does not help any more. The reason is that almost all the outliers (*TF-outlier* or *MO-outlier*) can be found when ρ (or ι) = 1 or 2. It's worth mentioning that $OTODS$ identifies the outlierness of trajectory fragments by using ally fragments, but the validity of ally fragment and local micro-cluster structures ensures that $OTODS$ detects almost all the outliers, and thus achieves a quite high recall rate.

Third, we examine $F\text{-}measure$ of $TODS$ and $OTODS$ algorithms under different thresholds, as shown in Fig. 12. We observe that both algorithms obtain the best $F\text{-}measure$ for *TF-outlier* detection when $\rho = 2$ and d (or d_a)≥ 0.0032, and for *MO-outlier* detection when $\iota = 2$ and d (or d_a) = 0.0032. This is due to that the average pairwise distance of trajectory fragments is about 0.0032 on $TaxiShanghai13$, and at the same time LAFs of most *TF-outlier*s and EAFs of *MO-outlier*s reach about 2. From the above results, we conclude that our proposal can be tolerate to noisy disturbance and has a lower false alarm rate on account of evolutionary anomaly assessment. Furthermore, $OTODS$ can attain almost the same effectiveness as $TODS$ as long as the thresholds are set appropriately.

5.3 Efficiency Evaluation

In this section we conduct a study to assess the efficiency of our proposal by comparing $TODS$ with $OTODS$ when dealing with streaming trajectories. The number of trajectories gradually grows from $40\,\mathrm{k}$ to $280\,\mathrm{k}$. Fig. 13a shows the

execution time comparison (expressed as seconds) between $TODS$ and $OTODS$ upon $TaxiShanghai13$. The execution time of both algorithms scale linearly with data size, and $OTODS$ keeps superior to that processed by $TODS$ with the progression of trajectory stream. The faster processing rate of $OTODS$ is contributed by the structures of ally fragment and local micro-cluster. That is, we estimate local anomaly factor of each ally fragment in each local micro-cluster instead of evaluate local anomaly factor for each trajectory fragment. More importantly, for $OTODS$ algorithm, along with the sliding forward of time window, whether insertion of new fragment or elimination of obsolete fragment only influences a small amount of ally fragments in certain local micro-clusters. Therefore, $OTODS$ algorithm significantly outperforms $TODS$ algorithm. This also demonstrates the time-quality trade-off by comparing $F\text{-}measure$ and execution time of both algorithms. $OTODS$ algorithm is capable of detecting the outliers upon streaming trajectories in a promising efficiency.

Moreover, in order to further test the robustness of $TODS$ and $OTODS$ algorithms, we study the sensitivity of both algorithms under different parameter settings. We run the comparative experiments by varying the values of d, N and d_a respectively. 1)*Varying d:* Because d is the distance threshold for determining the neighborhood of trajectory fragment, it is important to investigate its effect on the performance of $TODS$ algorithm. Fig. 13b shows the processing time comparison of $TODS$ when varying d from 0.0064 to 0.1024. There is a little difference in the comparison results of $TODS$ with different values, and $TODS$ attains the best efficiency when $d = 0.0064$. $TODS$ is almost insensitive to the value of d. The slight increase of execution time when the value of d increases is due to that more neighbors can be found within a larger d, and the computing costs of ldd and LAF accordingly increase. 2)*Varying N:* Fig. 13c shows the processing time comparison of $OTODS$ when varying N ($N = 120$ s, 300 s and 600 s respectively), $timebin$ is set as 30 s, 60 s and 120 s respectively. We observe that the processing time of $OTODS$ algorithm increases as the trajectory data continue to flow in, and it is modestly influenced by a larger window size. It is based on the fact that a larger window size leads to more incoming trajectories need to be simplified into fragments and then detected whether existing outliers. Even so, the processing time of $OTODS$ algorithm is only 59 s when $N = 600$ s. 3)*Varying d_a:* Fig. 13d shows the processing time comparison of $OTODS$ when varying d_a from 0.0002 to 0.0032. With the increase of trajectory data, the execution time increases accordingly, and $OTODS$ attains the best efficiency when $d_a = 0.0016$. This is due to the fact that a smaller d_a leads to substantial ally fragments, which increases the cost of searching nearest ally fragment for each incoming trajectory fragment and the computation overhead of ldd and LAF. From the above experimental results, we conclude that $TODS$ and $OTODS$ algorithms can effectively identify outliers upon trajectory stream, while $OTODS$ is more efficient than $TODS$.

6 Conclusion and Future Work

Existing outlier detection techniques on trajectory stream determine the outlierness of trajectories based upon spatial proximity relationship, they cannot discover the outliers which have numerous close neighbors, while behaving differently from their neighbors. In this paper, we first divide the features of trajectory into two groups (*Similarity Feature* and *Difference Feature*), and address the issue of online identifying local and evolutionary trajectory outlier. On the basis of that, we propose a framework to identify outliers upon streaming trajectories. It consists of two components, including trajectory simplification phase and outlier detection phase. We present a basic algorithm ($TODS$), and an optimized algorithm ($OTODS$) that incorporates a new data structure (*Ally Fragment*) to detect both types of outliers. We validate our proposal for effectiveness and efficiency by conducting extensive experiments on three real data sets, and show that our proposed algorithms are efficient in continuously detecting both types of outliers upon trajectory streams.

Trajectory data in emerging big data applications is more generally collected in a distributed fashion. With the substantial increment of trajectory stream data, a highly distributed trajectory outlier detection will become indispensable solution. Recently, due to the scalability, flexibility and cost effectiveness, numerous distributed computing platforms such as Spark and Storm have become prevalent. Therefore, for future work, we would like to extend $TODS$ algorithm to a distributed solution based upon distributed computing platform, to improve the efficiency of processing streaming trajectories and provide more efficient outlier detection results.

Acknowledgments

The authors are very grateful to the editors and reviewers for their valuable comments and suggestions. This work is supported by the National Key Research and Development Program of China (No.2016YFB1000905), NSFC (Nos.61702423, 61370101, 61532021, U1501252, U1401256, and 61402180), the Natural Science Foundation of the Education Department of Sichuan Province (No.17ZA0381), the China West Normal University Special Fundation of National Programme Cultivation (No.16C005), and the Meritocracy Research Funds of China West Normal University (No.17YC158).

References

[1] S. Shang, R. Ding, B. Yuan, K. Xie, K. Zheng, and P. Kalnis, "User oriented trajectory search for trip recommendation," in *Proc. 15th Int. Conf. Extending Database Technol.*, 2012, pp. 156–167.

[2] K. Zheng, S. Shang, N. J. Yuan, and Y. Yang, "Towards efficient search for activity trajectories," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2013, pp. 230–241.

[3] S. Shang, L. Chen, C. S. Jensen, J. Wen, and P. Kalnis, "Searching trajectories by regions of interest," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 7, pp. 1549–1562, Jul. 2017.

[4] H. Liu, C. Jin, and A. Zhou, "Popular route planning with travel cost estimation," in *Proc. Int. Conf. Database Syst. Advanced Appl.*, 2016, pp. 403–418.

[5] X. Duan, C. Jin, X. Wang, A. Zhou, and K. Yue, "Real-time personalized taxi-sharing," in *Proc. Int. Conf. Database Syst. Advanced Appl.*, 2016, pp. 451–465.

[6] T. Wang, J. Mao, and C. Jin, "Hymu: A hybrid map updating framework," in *Proc. Int. Conf. Database Syst. Advanced Appl.*, 2017, pp. 19–33.

[7] D. M. Hawkins, *Identification of Outliers*. Berlin, Germany: Springer, 1980.

[8] V. Barnett, *Outliers in Statistical Data*. Hoboken, NJ, USA: Wiley, 1994.

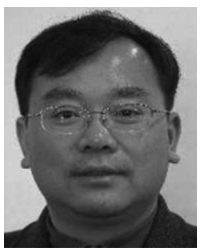
[9] P. Lei, "A framework for anomaly detection in maritime trajectory behavior," *Knowl. Inf. Syst.*, vol. 47, no. 1, pp. 189–214, 2016.

[10] J. Lee, J. Han, and X. Li, "Trajectory outlier detection: A partition-and-detect framework," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2008, pp. 140–149.
[11] Y. Ge, H. Xiong, C. Liu, and Z. Zhou, "A taxi driving fraud detection system," in *Proc. IEEE Int. Conf. Data Mining*, 2011, pp. 181–190.
[12] S. Liu, L. M. Ni, and R. Krishnan, "Fraud detection from taxis' driving behaviors," *IEEE Trans. Veh. Technol.*, vol. 63, no. 1, pp. 464–472, Jan. 2014.
[13] H. Wu, C. Tu, W. Sun, B. Zheng, H. Su, and W. Wang, "GLUE: A parameter-tuning-free map updating system," in *Proc. 18th ACM Conf. Inf. Knowl. Manage.*, 2015, pp. 683–692.
[14] Y. Zhu, Y. Wang, G. Forman, and H. Wei, "Mining large-scale GPS streams for connectivity refinement of road maps," *Comput. J.*, vol. 58, no. 9, pp. 2109–2119, 2015.
[15] J. Lee, J. Han, and K. Whang, "Trajectory clustering: A partition-and-group framework," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2007, pp. 593–604.
[16] Y. Bu, L. Chen, A. W. Fu, and D. Liu, "Efficient anomaly monitoring over moving object trajectory streams," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 159–168.
[17] Y. Yu, L. Cao, E. A. Rundensteiner, and Q. Wang, "Detecting moving object outliers in massive-scale trajectory streams," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 422–431.
[18] K. Zheng, Y. Zheng, N. J. Yuan, and S. Shang, "On discovery of gathering patterns from trajectories," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2013, pp. 242–253.
[19] K. Zheng, Y. Zheng, N. J. Yuan, S. Shang, and X. Zhou, "Online discovery of gathering patterns over trajectories," *Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1974–1988, 2014.
[20] S. Shang, R. Ding, K. Zheng, C. S. Jensen, P. Kalnis, and X. Zhou, "Personalized trajectory matching in spatial networks," *Int. J. Very Large Data Bases*, vol. 23, no. 3, pp. 449–468, 2014.
[21] J. Mao, Q. Song, C. Jin, Z. Zhang, and A. Zhou, "TSCluWin: Trajectory stream clustering over sliding window," in *Proc. Int. Conf. Database Syst. Advanced Appl.*, 2016, pp. 133–148.
[22] E. M. Knorr and R. T. Ng, "Algorithms for mining distance-based outliers in large datasets," in *Proc. 30th Int. Conf. Very Large Data Bases*, 1998, pp. 392–403.
[23] E. M. Knorr and R. T. Ng, "Finding intensional knowledge of distance-based outliers," in *Proc. 30th Int. Conf. Very Large Data Bases*, 1999, pp. 211–222.
[24] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-based outliers: Algorithms and applications," *Int. J. Very Large Data Bases*, vol. 8, no. 3–4, pp. 237–253, 2000.
[25] X. Li, Z. Li, J. Han, and J. Lee, "Temporal outlier detection in vehicle traffic data," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2009, pp. 1319–1322.
[26] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xing, "Discovering spatio-temporal causal interactions in traffic data streams," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 1010–1018.
[27] S. Chawla, Y. Zheng, and J. Hu, "Inferring the root cause in road traffic anomalies," in *Proc. IEEE Int. Conf. Data Mining*, 2012, pp. 141–150.
[28] B. Pan, Y. Zheng, D. Wilkie, and C. Shahabi, "Crowd sensing of traffic anomalies based on human mobility and social media," in *Proc. 21st ACM SIGSPATIAL Int. Conf. Advances Geographic Inf. Syst.*, 2013, pp. 334–343.
[29] J. Mao, C. Jin, Z. Zhang, and A. Zhou, "Anomaly detection for trajectory big data: Advancements and framework," *J. Softw.*, vol. 28, no. 1, pp. 17–34, 2017.
[30] X. Li, J. Han, S. Kim, and H. Gonzalez, "ROAM: rule- and motif-based anomaly detection in massive moving object data sets," in *Proc. 7th SIAM Int. Conf. Data Mining*, 2007, pp. 273–284.
[31] W. Yang, Y. Gao, and L. Cao, "TRASMIL: A local anomaly detection framework based on trajectory segmentation and multi-instance learning," *Comput. Vis. Image Understanding*, vol. 117, no. 10, pp. 1273–1286, 2013.
[32] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2000, pp. 427–438.
[33] Y. Ge, H. Xiong, Z. Zhou, H. T. Ozdemir, J. Yu, and K. C. Lee, "Top-eye: Top-k evolving trajectory outlier detection," in *Proc. 18th ACM Conf. Inf. Knowl. Manage.*, 2010, pp. 1733–1736.
[34] D. Zhang, N. Li, Z. Zhou, C. Chen, L. Sun, and S. Li, "iBAT: Detecting anomalous taxi trajectories from GPS traces," in *Proc. 12th ACM Int. Conf. Ubiquitous Comput.*, 2011, pp. 99–108.
[35] C. Chen, et al., "iBOAT: Isolation-based online anomalous trajectory detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 806–818, Jun. 2013.
[36] C. Chen, et al., "Real-time detection of anomalous taxi trajectories from GPS traces," in *Proc. Int. Conf. Mobile Ubiquitous Syst.: Comput. Netw. Serv.*, 2011, pp. 63–74.
[37] M. M. Breunig, H. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2000, pp. 93–104.
[38] G. Cormode, V. Shkapenyuk, D. Srivastava, and B. Xu, "Forward decay: A practical time decay model for streaming systems," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2009, pp. 138–149.
[39] N. Pelekis, I. Kopanakis, E. E. Kotsifakos, E. Frentzos, and Y. Theodoridis, "Clustering trajectories of moving objects in an uncertain world," in *Proc. IEEE Int. Conf. Data Mining*, 2009, pp. 417–427.
[40] A. Anagnostopoulos, M. Vlachos, M. Hadjieleftheriou, E. J. Keogh, and P. S. Yu, "Global distance-based segmentation of trajectories," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2006, pp. 34–43.
[41] C. Panagiotakis, N. Pelekis, I. Kopanakis, E. Ramasso, and Y. Theodoridis, "Segmentation and sampling of moving object trajectories based on representativeness," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 7, pp. 1328–1343, Jul. 2012.

Jiali Mao is an associate professor with China West Normal University. She is currently working toward the PhD degree in the School of Data Science and Engineering, East China Normal University. Her research interests include location-based services and spatio-temporal data mining.

Tao Wang is working toward the graduate degree in the School of Computer Science and Software Engineering, East China Normal University. His research interests include spatio-temporal data mining and location-based services.

Cheqing Jin received the bachelor's and master's degrees from Zhejiang University, and the PhD degree from Fudan University. He is a professor with East China Normal University. His research interests include streaming data management, location-based services, uncertain data management, data quality, and database benchmarking. He is a member of the IEEE.



Aoying Zhou is a professor with East China Normal University. He is now acting as a vice-director of the ACM SIGMOD China and Database Technology Committee of the China Computer Federation. He is serving as a member of the editorial boards of the *VLDB Journal*, *WWW Journal*, etc. His research interests include data management for data-intensive computing, and memory cluster computing. He is a member of the IEEE.