# Sub-trajectory- and Trajectory-Neighbor-based Outlier Detection over Trajectory Streams

Zhihua Zhu[1,2], Di Yao[1,2], Jianhui Huang[1], Hanqiang Li[3], and Jingping Bi[1(✉)]

[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
{zhuzhihua,yaodi,huangjianhui,jpingbi}@ict.ac.cn
[2] University of Chinese Academy of Sciences, Huairou, China
[3] National Defence Key Laboratory of Blind Processing of Signals, Chengdu, China
cnpla@126.com

**Abstract.** Precisely and efficiently anomaly detection over trajectory streams is critical for many real-time applications. However, due to the uncertainty and complexity of behaviors of objects over trajectory streams, this problem has not been well solved. In this paper, we propose a novel detection algorithm, called STN-Outlier, for real time applications, where a set of fine-grained behavioral features are extracted from the sub-trajectory instead of point and a novel distance function is designed to measure the behavior similarity between two trajectories. Additionally, an optimized framework(TSX) is introduced to reduce the CPU resources cost of STN-Outlier. The performance experiments demonstrate that STN-Outlier successfully captures more fine-grained behaviors than the state-of-the-art methods; besides, the TSX framework outperforms the baseline solutions in terms of the CPU time in all cases.

**Keywords:** Outlier · Sub-trajectory · Trajectory streams

## 1 Introduction

Today, the location-acquisition devices such as GPS and smart phone, monitoring the behaviors of vehicles and people, are generating massive-scale high-speed trajectory streams. The applications like traffic management and security surveillance, need continuously detect the abnormal objects from high volumes of objects in such heavy data. Those abnormal objects such as drunk drive in traffic management or espionage in security surveillance, whose erratic behaviors are different from the majority in trajectory streams, must be detected efficiently based on the behaviors over a period of time and reported in time. Even a short time delay may lead to losses of huge funds.

This work has been supported by the National Natural Science Foundation of China (No. 61472403 and 61702470) and the Beijing Natural Science Foundation (No. 4182062).


D. Phung et al. (Eds.): PAKDD 2018, LNAI 10937, pp. 551–563, 2018.
https://doi.org/10.1007/978-3-319-93034-3_44

In existing studies, part of researchers use the techniques that normally are machine learning to get the discriminative model using the global characteristics of the dataset [3–8]. However, due to the concept drift in trajectory stream, which means the behaviors of the target object change over time in unforeseen ways, using one single pre-trained model to continuously detect outliers would lead to inaccurate results; besides, rebuilding the model periodically would result in expensive modeling costs for real-time applications. In addition, in a trajectory stream populated with massive scale moving objects, the moving patterns of high volumes of moving objects are more dynamic and complex than single object. That is, the moving pattern of one object over trajectory streams with a lot of objects is not suit to be modeled by the local continuity assumption [1,2,9]. Considering the above challenges, Yu et al. [6] proposed a novel method called TN-Outlier to detect the trajectory outliers over a real-time trajectory stream. However, the work only used the spatial distance between trajectory points to evaluate the relationships among trajectories. As a result, it is difficult to distinguish the difference of behaviors of moving objects especially when they are close to each other. Although the importance of continuously detecting such types of outliers, this problem has not been well solved.

Because distance-based outlier is robust against concept drift and amenable to swiftly evicting obsolete models of outlierness [16]; besides, many works [4,13,14] used sub-trajectories instead of points to measure the trajectory similarity and achieved better results, but they only considered the directional and spatial differences between two sub-trajectories which is hard to capture the detail differences of behaviors. In this paper, we propose a novel distance-based outlier definition, called Sub-trajectory- and Trajectory-Neighbor-based trajectory Outlier(STN-Outlier), to detect the complex abnormal behaviors in a trajectory stream. The definition not only considers the behavioral approximation of moving objects in a region, but also takes the duration of the behavioral similarity across time into account. Specially, a novel distance function that combines both the inter- and intra-trajectory features in an integrated manner is designed to measure the fine-grained difference between two trajectories. Then, a comprehensive framework, called the *temporal and spatial-aware examination* (TSX), is introduced to efficiently detect the outliers over high volume trajectory stream. The experimental studies on synthetic and real Taxi [6,10,11] datasets demonstrate that the STN-Outlier successfully captures the deviating behaviors effectively over other state-of-the-art methods; besides, the TSX framework outperforms the baseline solutions in terms of the CPU time in all cases.

## 2 Overview

### 2.1 Preliminary

We define the scenario of trajectory outlier detection. $O = \{o_1, o_2, \ldots, o_n\}$ denotes a dataset composed of $n$ observed moving objects. Each moving object $o_m \in O$ is represented as an infinite sequence of trajectory points $Tr_m = \{p_m^1, p_m^2, \ldots p_m^i, \ldots\}$ at timebins $\{t_1, t_2, \ldots t_i, \ldots\}$, where the term "timebin" is

referred to the smallest time granularity that ensures each trajectory has at least one point fall into each bin. Given $n$ moving objects, a trajectory stream $S$ is represented as $n$ infinite sequences of trajectory points ordered by timebins $S = \{p_1^1 p_2^1 \dots p_n^1, p_1^2 p_2^2 \dots p_n^2, \dots, p_1^i p_2^i \dots p_n^i, \dots\}$, where $p_1^i p_2^i \dots p_n^i$ are said to fall into the same timebin $i$ in $S$. Then, a periodic sliding window $W$ with a fixed window size $w$ and slide length $s$ is used to extract a finite sub-stream for processing.

Additionally, we use the tuple $(p_m^i, p_m^{i+1}, \boldsymbol{f}_{s_m^{i,i+1}})$ to represent a sub-trajectory $s_m^{i,i+1}$ of $Tr_m$, where $\boldsymbol{f}_{s_m^{i,i+1}}$ is a feature vector collected by other devices.

### 2.2 Problem Formulation

In Fig. 1, according to [6] $Tr_6$ is obviously recognized as an outlier since it changes its neighbors frequently. On the contrary, $Tr_1$ and $Tr_2$ would be detected as inliers, if they keep being neighbors, namely the Euclidean distances between their points are less than $d$ (a distance threshold) at all 6 timebins. Likewise, $Tr_{3-5}$ would be labeled as inliers *w.r.t* the same $d$. However, note that $Tr_3$ whose behavior is completely different from others should be detected as outlier not an inlier even though its points are close to the points of $Tr_4$ and $Tr_5$.

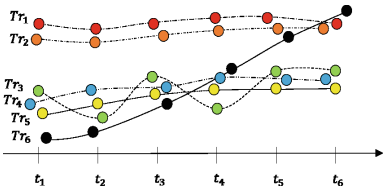



**Fig. 1.** Six trajectories in a window with size $w = 6$. Outlier: $Tr_6$, $Tr_3$; Inlier: $Tr_1, Tr_2, Tr_4, Tr_5$

To describe the trajectory outliers, e.g. $Tr_3$ in Fig. 1, classes of novel notions of distance-based outliers are defined referring to the semantics [6].

**Definition 1 (*Sub-trajectory Neighbor*).** *Given two sub-trajectories $s_m^{i,i+1}$ and $s_n^{i,i+1}$, they are said to be **sub-trajectory neighbors** if $dist(s_m^{i,i+1}, s_n^{i,i+1}) \leq d$ where $dist(s_m^{i,i+1}, s_n^{i,i+1})$ is a distance function and $d$ is a distance threshold.*

The ***sub-trajectory neighbor set*** between $Tr_m$ and $Tr_n$ *w.r.t* a distance threshold $d$ in a $W$ is denoted as $\mathbb{N}_{mn}^d$, with $|\mathbb{N}_{mn}^d|$ denoting the size.

**Definition 2 (*Trajectory Neighbor*).** *In window $W$, given a distance threshold $d$ and timebin count threshold $thr_t$, trajectory $Tr_m$ is called a **trajectory neighbor** of $Tr_n$ if $|\mathbb{N}_{mn}^d| \geq thr_t$.*

**Definition 3 (*Trajectory Outlier*).** *Given a distance threshold $d$, a neighbor count threshold $k$, and timebin count threshold $thr_t$, a trajectory $Tr_m$ in the window $W$ is a* ***trajectory-neighbor based trajectory outlier*** *if $Tr_m$ has at most $k-1$ trajectory neighbors in $W$ with trajectory neighbor as per Definition 2.*

Given the parameters $d$, $k$, $thr_t$ and $n$ trajectories in window $W$, *our goal is to detect and report all trajectory-neighbor based trajectory outliers in the window $W$ with high accuracy and efficiency.*

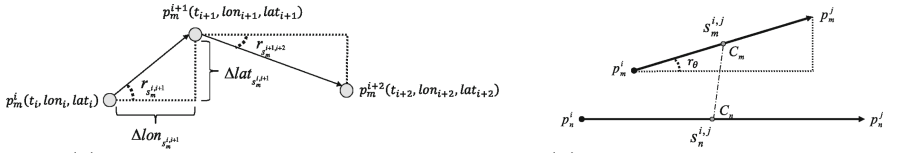
## 3 Methodology

In this section, we introduce a novel distance function for sub-trajectories. The key of this function is the combination of two types of features that ensures a significant difference between outliers and inliers can be captured.

### 3.1 Feature Extraction

To describe the difference of behaviors of trajectories minutely, two types of features intra- and inter-trajectory features are extracted for each sub-trajectory.

**Intra-trajectory Feature.** Intra-trajectory features indicate the moving behavior of each trajectory that can be quantized by the differences of attributes between two consecutive trajectory points.




(a) The intra-trajectory featurese (b) The inter-trajectory features

**Fig. 2.** The elements for intra- and inter-trajectory features

Let us consider a window with $n$ 2-dimensions points belonging to $Tr_m$. Specifically, suppose the point is GPS point. Each point $p_m^i$ can be denoted as $(t_i, lon_{p_m^i}, lat_{p_m^i})$ where $t_i$ is the $i$th timebin, $lon_{p_m^i}$ and $lat_{p_m^i}$ denote the longitude and latitude respectively. For sub-trajectory $s_m^{i,i+1}$, four attributes that intuitively describe the behaviors of one trajectory are extracted as the intra-trajectory features. As shown in Fig. 2(a), the features are change of longitude $\Delta lon_{s_m^{i,i+1}} = lon_{p_m^{i+1}} - lon_{p_m^i}$, change of latitude $\Delta lat_{s_m^{i,i+1}} = lat_{p_m^{i+1}} - lat_{p_m^i}$, speed $v_{s_m^{i,i+1}} = \sqrt{(\Delta lon_{s_m^{i,i+1}})^2 + (\Delta lat_{s_m^{i,i+1}})^2} / (t_{i+1} - t_i)$ and change of rate of turn(ROT) $\Delta r_{s_m^{i,i+1}} = r_{s_m^{i,i+1}} - r_{s_m^{i+1,i+2}}$, where $r_{s_m^{i,i+1}} = arctan(\Delta lat_{s_m^{i,i+1}} / \Delta lon_{s_m^{i,i+1}})$. Specially, for the last sub-trajectory $s_m^{n-1,n}$ of $Tr_m$, we set $\Delta r_{s_m^{n-1,n}} = 0$.

In order to intuitively reflect the moving direction change of a trajectory, attribute $\hat{d}_{s_m^{i,i+1}}$ is defined to flag whether object changes its moving direction between two consecutive sub-trajectories or not. Given the moving direction of $s_m^{i+1,i+2}$ as the standard direction, $\hat{d}_{s_m^{i,i+1}} = 0$ if $|\Delta r_{s_m^{i,i+1}}| \leq \pi/4$ and $\hat{d}_{s_m^{i,i+1}} = 1$ otherwise. After computing these features for each sub-trajectory, we get a feature vector $\boldsymbol{f}_{s_m^{i,i+1}} = (\Delta lon_{s_m^{i,i+1}}, \Delta lat_{s_m^{i,i+1}}, v_{s_m^{i,i+1}}, \Delta r_{s_m^{i,i+1}}, \hat{d}_{s_m^{i,i+1}})$.

**Inter-trajectory Feature.** Inter-trajectory features reflect the spatial difference and directional difference of two trajectories. It ensures the outliers that always moves alone or always in the areas where other objects rarely visit can be detected. The inter-trajectory features consist of two components: the spatial distance $d_c$ and angle distance $d_\theta$. Suppose there are two sub-trajectories $s_m^{i,i+1}$ and $s_n^{i,i+1}$, and their elements are intuitively illustrated in Fig. 2(b).

The spatial distance between $s_m^{i,i+1}$ and $s_n^{i,i+1}$ is denoted as the Euclidean distance between center points of $s_m^{i,i+1}$ and $s_n^{i,i+1}$, namely $d_c(s_m^{i,i+1}, s_n^{i,i+1}) = \|C_m - C_n\|$. The angle distance between $s_m^{i,i+1}$ and $s_n^{i,i+1}$ is defined as Formula 1. It is the intersection angle between $s_m^{i,i+1}$ and $s_n^{i,i+1}$. Here, $\|s_m^{i,i+1}\|$ denotes the length of $s_m^{i,i+1}$.

$$d_\theta(s_m^{i,i+1}, s_n^{i,i+1}) = \begin{cases} arccos(\frac{s_n^{i,i+1} \cdot s_m^{i,i+1}}{\|s_m^{i,i+1}\| \|s_n^{i,i+1}\|}) & \text{if } \|s_m^{i,i+1}\| \neq 0 \ \& \ \|s_n^{i,i+1}\| \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

### 3.2 Distance Function

Now based on the features denoted above, the distance function is defined to measure the similarity between $s_m^{i,i+1}$ and $s_n^{i,i+1}$. The function consists of two terms: similarity measure $sim(\boldsymbol{f}_{s_m^{i,i+1}}, \boldsymbol{f}_{s_n^{i,i+1}})$ and punishing item $\omega(d_c, d_\theta)$. Namely:

$$dist(s_m^{i,i+1}, s_n^{i,i+1}) = 1 - \omega(d_c, d_\theta) \times sim(\boldsymbol{f}_{s_m^{i,i+1}}, \boldsymbol{f}_{s_n^{i,i+1}}) \quad (2)$$

In particular, considering the differences of intra-trajectory features in semantic and value, we need a normalization function to normalize the features. However, due to the uncertainty caused by the concept drift over continuous trajectory streams, the minimum or maximum of each feature cannot be fixed. It is not suit to use the Min-Max scaling to normalize the features. Likewise, the Z-score standardization cannot be used.

Given two feature vectors $\boldsymbol{f}_{s_m^{i,i+1}}$, $\boldsymbol{f}_{s_n^{i,i+1}}$, a ratio $\rho^j = |f^j_{s_m^{i,i+1}} - f^j_{s_n^{i,i+1}}| \,/\, max\{f^j_{s_m^{i,i+1}}, f^j_{s_n^{i,i+1}}\}$ is computed to normalize the differences between features into $[0, 1]$, where $f^j$ is the $j$th feature in feature vector. However, when $f^j \in \{\Delta lon, \Delta lat, \Delta r\}$, if one of $f^j_{s_m^{i,i+1}}$ and $f^j_{s_n^{i,i+1}}$ is equal to 0, the $\rho^j$ would be amplified to 1 even if their difference is small; otherwise, we expect a small difference amplify to 1 when one of objects is static. Therefore, we use $\sigma = sin(d_\theta(s_m^{i,i+1}, s_n^{i,i+1}))$ as the constraint factor to limit the value of $\rho^j$. That is, the smaller $d_\theta$ is, the smaller $\rho^j$ is. The new ratio $\hat{\rho}^j$ is defined as follows.

If $v_{s_m^{i,i+1}} = 0$ or $v_{s_n^{i,i+1}} = 0$, then $\hat{\rho}^j = \rho^j$. And if $v_{s_m^{i,i+1}} \neq 0$, $v_{s_n^{i,i+1}} \neq 0$ and $f^j \in \{\Delta lon, \Delta lat, \Delta r\}$, then $\hat{\rho}^j = \rho^j \times \sigma$.

Specially, for $max\{f^j_{s_m^{i,i+1}}, f^j_{s_n^{i,i+1}}\} = 0$, we set $\rho^j = 0$. Based on the formulas above, a new vector $\hat{\boldsymbol{\rho}} = (\hat{\rho}_{\Delta lon}, \hat{\rho}_{\Delta lat}, \hat{\rho}_v, \hat{\rho}_{\Delta r}, \hat{\rho}_{\hat{d}})$ is computed for each sub-trajectory pair. Then, to map the similarity between two sub-trajectories into $[0, 1]$, the function $sim(\boldsymbol{f}_{s_m^{i,i+1}}, \boldsymbol{f}_{s_n^{i,i+1}})$ is defined as

$$sim(\boldsymbol{f}_{s_m^{i,i+1}}, \boldsymbol{f}_{s_n^{i,i+1}}) = 1 - \frac{\|\hat{\boldsymbol{\rho}}\|_2}{\sqrt{|\hat{\boldsymbol{\rho}}|}} \tag{3}$$

where $\|\cdot\|_2$ denotes the $L2$ norm and $|\hat{\boldsymbol{\rho}}|$ is the size of $\hat{\boldsymbol{\rho}}$. It means that the more similar $\boldsymbol{f}_{s_m^{i,i+1}}$ and $\boldsymbol{f}_{s_n^{i,i+1}}$ are, the bigger value of $sim(\boldsymbol{f}_{s_m^{i,i+1}}, \boldsymbol{f}_{s_n^{i,i+1}})$ is.

Next, considering the influence of spatial distance and moving direction on similarity comparison we define a punishing item $\omega(d_c, d_\theta)$ to control the value of $sim(s_m^{i,i+1}, s_n^{i,i+1})$. Given a spatial distance limit $\xi$, the punishing item $\omega(d_c, d_\theta)$ is defined as:

$$\omega(d_c, d_\theta) = \begin{cases} e^{-\frac{|d_c - \xi|}{\xi}} \times cos(d_\theta) & \text{if } d_c > \xi \\ cos(d_\theta) & \text{otherwise} \end{cases} \tag{4}$$

Namely, given the spatial distance threshold $\xi$ set by user, the farther apart two sub-trajectories the less similar they are. Likewise, the bigger difference of moving direction between two sub-trajectories the less similar they are.

## 4 Detection Framework

### 4.1 The Basic Framework

The basic framework of our trajectory outlier detection is shown as Fig. 3. In the basic framework, the STN-Outlier detects the outliers by first running a range query search for each trajectory at current window. The time complexity is $O(n)$ where $n$ is the number of trajectories in current window. Then, it traverses all neighbor of $Tr_m$ to determine the status of $Tr_m$, of which the worst case is to traverse all $n - 1$ trajectories. Thus, its worst complexity is $O(wn^2)$ where $w$ is window size. And, it would fully reuses the neighbor relationships collected in the previous window, there is high computational costs when $n$ is large.

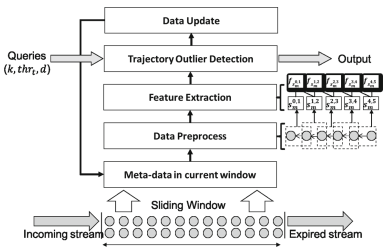



**Fig. 3.** The basic framework of STN-Outlier

Inspired by the minimal examination (MEX) framework [6] and the locality-sensitive hashing (LSH) algorithm [16], we design a temporal and spatial-aware examination (TSX) framework that efficiently solves this issue.

### 4.2 The Optimized Framework

A trajectory $Tr_m$ will be labeled based on the neighbor evidence that has been acquired for this object. Note that those neighbors must be near to $Tr_m$ in space. If a trajectory is far away from $Tr_m$, namely exceeding a radius $R$, then the trajectory must be non-neighbor of $Tr_m$. This fact leads to an important observation. That is, to identify whether a trajectory $Tr_m$ is a neighbor-based inlier, it is unnecessary to compare with all other trajectories in window $W$. Instead a subset of the full trajectories that are near to $Tr_m$ often can be sufficient to prove that it is an inlier. To acquired the small yet sufficient subset of trajectories for $Tr_m$ a concept of *R-near Subset* with LSH is defined.

Specially, in TSX framework, the location of a trajectory is represented as an average of full points in the current window. Namely, the location $l_m$ of a trajectory $Tr_m$ in window $W$ is denoted as $\frac{\sum_{j=1}^{w} p_m^j}{w}$, where $W = \{p_m^1, p_m^2, \ldots, p_m^w\}$.

**Definition 4 (*R-near Subset*).** *Given an LSH family $\mathcal{F}$ and a trajectory set $DB_{Tr}$ in a window $W$, for a trajectory $Tr_m \in DB_{Tr}$, its R-near Subset $TR_{Tr_m}$ is denoted as $\{Tr_n | h(l_n) = h(l_m), h \in \mathcal{F}, Tr_n \in DB_{Tr}\}$ with a collision probability at least $1 - \delta$, which denotes the probability that $Tr_m, Tr_n$ collide for a hash function uniformly chosen from the family $\mathcal{F}$.*

In particular, we choose $E^2LSH$ [17] that is defined for the case where the distances are measured according to the Euclidean norm to solve the R-near Subset problem. In $E^2LSH$, a new family $\mathcal{G}$ of hash functions $g$ is defined. Each function $g$ is obtained by concatenating $K$ functions $h_1, \ldots, h_K$ from $\mathcal{F}$, i.e., $g(p) = [h_1(p), \ldots, h_K(p)]$. Finally, the algorithm constructs $L$ hash tables, each corresponding to a different randomly chosen hash function $g$. Based on the definitions and theory analysis for $E^2LSH$, we get the following lemma.

**Lemma 1.** *Given a $E^2LSH$ family $\mathcal{G}$ and $L$ corresponding hash tables, a trajectory $Tr_m$ could find a R-near subset $TR_{Tr_m} = \{Tr_n | g_j(l_m) = g_j(l_n), g_j \in \{g_1, \ldots, g_L\}\}$ with a collision probability $1 - \delta \geq 0.9$ iff $L \geq \frac{log10}{-log(1-P_1^K)}$ for a fixed $K$ and $P_1$, where $P_1 = p(R) = Pr_{h \in \mathcal{F}}[h(p) = h(q)] = 1 - 2norm(-b/R) - \frac{2}{\sqrt{2\pi}b/R}(1 - e^{-b^2/2R^2})$ that $b$ is the size of bucket.*

*Proof.* Consider a query trajectory $Tr_m$ and an R-near trajectory $Tr_n$ of $Tr_m$. According to the definitions of $E^2LSH$, we get $Pr_{g \in \mathcal{G}}[g(q) = g(p)] \geq P_1^K$. Thus, $Tr_m$ and $Tr_n$ fail to collide for all $L$ functions $g_j$ with probability at most $(1 - P_1^K)^L$. Requiring that the trajectory $Tr_m$ collides with $Tr_n$ on some function $g_j$ is equivalent to saying $1 - (1 - P_1^K)^L \geq 1 - \delta$. Therefore, given a fixed $K$ and $P_1$, if setting $1 - \delta \geq 0.9$ namely $\delta \leq 0.1$, then we get: $1 - (1 - P_1^K)^L \geq 0.9 \Leftrightarrow (1 - P_1^k)^L \leq 0.1 \Leftrightarrow L \times log(1 - P_1^K) \leq log(1/10) = -log10 \Leftrightarrow L \geq \frac{log10}{-log(1-P_1^K)}$.

Lemma 1 shows the effectiveness of R-near Subset that guarantees the effectiveness and efficiency of STN-Outler. with the R-near subset, we are ready to propose one spatial-aware principle for optimizing the MEX framework.

**Algorithm 1.** STN-Outlier using TXS framework

```
   Input: Trajectory Set DB_Tr, the current window W_c, hash tables H, a E^2LSH family G,
          parameters: d, k, and thr_t
   Output: Outliers
1  DB_lt = []
2  for each Tr_i ∈ DB_Tr do
3      if Tr_i.lifetime ≤ W_c.start then
4          DB_lt ← Tr_i
5  for each Tr_i ∈ DB_lt do
6      for each Tr_j ∈ Tr_i.NT do
7          Time-aware Examination for Tr_i and Tr_j
8          Minimal Support examination for Tr_i
9      if |Tr_i.Neighbors| < k then
10         TR_Tr_m ← queryRnearSubset(H, G, Tr_m)
11         for each Tr_j ∈ (TR_Tr_m) do
12             Time-aware Examination for Tr_i and Tr_j
13             Minimal Support examination for Tr_i
14         if |Tr_i.getNeighbors| < k then
15             Tr_i is "outlier"
16     updating the lifetime of Tr_i
```

PRINCIPLE 1. ***Spatial-aware examination***: *Given a query $Q$ and the trajectory set $DB_{Tr}$ in the current window $W_c$, for evaluating a trajectory $Tr_m$, the examination principle suggests that a R-near subset $TR_{Tr_m}$ can replace the full trajectories $DB_{Tr}$ to determination the status of $Tr_m$.*

This principle aims to prove the status of a given trajectory $Tr_m$ by only discovering its R-near trajectories instead of searching through full trajectories. For space reasons, we omit the three principles(lines 2–4, 7 and 8 in Algorithm 1) of MEX framework and their proofs. See [6] for detail principles and proofs.

The new trajectory outlier detection algorithm is shown as Algorithm 1. The status of a trajectory will be re-examined when its lifetime expires(lines 2−4). And whenever a trajectory is being re-examined, the minimal support examination(lines 8 and 13) will cooperate with time-aware examination(lines 7 and 12) to re-establish the minimal support(=$k$) in its R-near subset(line 10). It does so by only acquiring enough new evidence rather than building a new minimal support from scratch.

## 5 Experiments

All experiments are performed on a server with Intel Xeon CPU 2.10 GHz. In our implementation, we simulate the streaming manner by using a sliding window in the main memory.

**Datasets.** The experiments are performed on synthetic datasets and real taxi data [6]. The real taxi data contains 1k trajectories and a outlier set manually labeled by a user study. However, in the user study the behavior of a taxi

driver is classified as abnormal by users only if he always operates in areas that other drivers rarely visit. Therefore, we apply two transformations [18], namely Add Noise Transformation and Random Shift Transformation, to the original trajectories and generate a series of new outliers to enrich the types of outliers.

**Metrics and Measurements.** For the effectiveness evaluation, we measure the quality of reported outliers by F1-Measure, at which $Precision = \frac{|R_0 \cap D_0|}{|D_0|}$ and $Recall = \frac{|R_0 \cap D_0|}{|R_0|}$ where $R_0$ denotes the set of outliers and $D_0$ is the outliers detected by the algorithm.

For the efficiency evaluation, we measure the CPU resources cost by recording the cost of the first window, at which each trajectory searches its neighbors over all trajectories and compares at least $thr_t$ segments with other trajectories.

### 5.1 Effectiveness Evaluation

We generate a stream with a controlled number of outliers. Specifically, 200 normal trajectories are first randomly sampled from the real taxi dataset. A random offset of radius $R = 5$ m is added to each sampled trajectory to emulate 3–10 trajectories. Then, 100 trajectories are randomly chosen from the sampled trajectories. Half of them are transformed by Add Noise Transformation, while the other half are transformed by Random Shift Transformation.

As shown in Table. 1, we evaluate the performance of STN-Outlier in comparison with TN-Outlier and a number of well-known sub-trajectory-based similarity measures, including CTraStream [14] and Hausdorff [13]. We execute

**Table 1.** Performance comparison by three evaluation metrics.

| $w = 15$ | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Method | | $k = 1$ | | | $k = 4$ | | | $k = 8$ | | |
| | | Precision | Recall | F1 | Precision | Recall | F1 | Precision | Recall | F1 |
| STN-Outlier | $thr_t = 8$ | 0.9177 | 0.9602 | **0.9385** | 0.5488 | 0.9668 | **0.7002** | 0.1702 | 0.9867 | **0.2904** |
| | $thr_t = 11$ | 0.7956 | 0.9801 | **0.8783** | 0.4639 | 0.9801 | **0.6297** | 0.1576 | 0.9867 | **0.2718** |
| TN-Outlier | $thr_t = 8$ | 1.0 | 0.3907 | 0.5619 | 0.4957 | 0.3907 | 0.4370 | 0.1442 | 0.6688 | 0.2373 |
| | $thr_t = 11$ | 1.0 | 0.4304 | 0.6018 | 0.5038 | 0.4304 | 0.4642 | 0.1478 | 0.6953 | 0.2439 |
| STN-CTraStream | $thr_t = 8$ | 1.0 | 0.3576 | 0.5268 | 0.45 | 0.3576 | 0.3985 | 0.1443 | 0.6423 | 0.2357 |
| | $thr_t = 11$ | 1.0 | 0.3841 | 0.5550 | 0.3295 | 0.3841 | 0.3547 | 0.1380 | 0.6754 | 0.2292 |
| STN-Hausdorff | $thr_t = 8$ | 0.1604 | 0.4569 | 0.2375 | 0.0576 | 0.5430 | 0.1041 | 0.0747 | 0.7549 | 0.1359 |
| | $thr_t = 11$ | 0.1579 | 0.5629 | 0.2467 | 0.0585 | 0.5695 | 0.1061 | 0.0742 | 0.7682 | 0.1353 |
| $w = 30$ | | | | | | | | | | |
| Method | | $k = 1$ | | | $k = 4$ | | | $k = 8$ | | |
| | | Precision | Recall | F1 | Precision | Recall | F1 | Precision | Recall | F1 |
| STN-Outlier | $thr_t = 15$ | 0.9423 | 0.9735 | **0.9576** | 0.5424 | 0.9735 | **0.6966** | 0.1710 | 0.9801 | **0.2913** |
| | $thr_t = 22$ | 0.8333 | 0.9933 | **0.9063** | 0.5 | 0.9933 | **0.6651** | 0.1651 | 0.9933 | **0.2832** |
| TN-Outlier | $thr_t = 15$ | 1.0 | 0.3509 | 0.5196 | 0.4690 | 0.3509 | 0.4015 | 0.1389 | 0.6357 | 0.2280 |
| | $thr_t = 22$ | 1.0 | 0.4105 | 0.5821 | 0.4920 | 0.4105 | 0.4476 | 0.1456 | 0.6887 | 0.2404 |
| STN-CTraStream | $thr_t = 15$ | 1.0 | 0.3377 | 0.5049 | 0.4690 | 0.3509 | 0.4015 | 0.1403 | 0.6357 | 0.2299 |
| | $thr_t = 22$ | 1.0 | 0.3576 | 0.5268 | 0.4782 | 0.3642 | 0.4135 | 0.1434 | 0.6688 | 0.2362 |
| STN-Hausdorff | $thr_t = 15$ | 0.184 | 0.4569 | 0.2623 | 0.0554 | 0.5298 | 0.1003 | 0.0723 | 0.7417 | 0.1319 |
| | $thr_t = 22$ | 0.1611 | 0.5496 | 0.2492 | 0.0596 | 0.5827 | 0.1081 | 0.0744 | 0.7748 | 0.1358 |

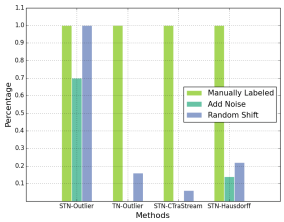



**Fig. 4.** The performance of methods on detecting three types of outlier

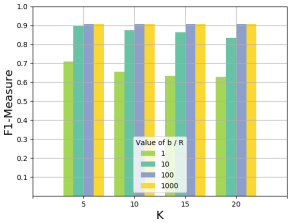



**Fig. 5.** The change of F1 with varying $K$ and $b/R$

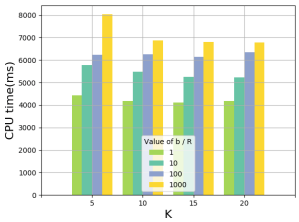



**Fig. 6.** The change of CPU time with varying $K$ and $b/R$

multiple queries that vary the parameters $k, thr_t$ and $w$ to study how the metrics is impacted in parameter space. Referring the experimental parameters setting in [6], the distance threshold $d$ is fixed as 0.1 and 300 m (=$\xi$) for STN-Outlier and other methods respectively. Besides, we set the weight set of Hausdorff to $(1, 1, 1)$ after tuning.

From Table. 1, STN-Outlier shows the better performance than other methods in F1-Measure, even if the inlier criteria is vary strict. In other words, STN-Outlier not only detects more outliers than others but also guarantees less false alarms. Furthermore, we study the performance of methods on detecting three types of outliers. When the inlier criteria is most relaxed where $k = 1, thr_t = 8$ and $w = 15$, the results are shown as Fig. 4. The most of outliers detected by TN-Outlier or CTraStream are manually labeled outliers, of which the detecting probability is near to 100%, while the outliers generated by the transformations are rarely or not detected. This is because the distance function in TN-Outlier or CTraStream is less focused on the differences of behaviors. In summary, compared with STN-Outlier, the TN-Outlier and other methods are worse in capturing and modeling more complex abnormal behaviors.

### 5.2 Efficiency Evaluation

Next we evaluate the efficiency of STN-Outlier with TSX. We vary the most important parameters, to (1) assess the impact of TSX framework versus the baseline, (2) evaluate sensitivity of parameter variations on STN-Outlier.

**Varying Parameters of LSH.** In this scenario, we vary the thresholds $K$ and $b/R(P_1)$ to study how F1-Measure and CPU time are impacted. The other parameters are fixed as $k = 1, thr_t = 22, w = 30, R = \xi = 300$ and $d = 0.1$.

Figure 5 shows that the F1-Measure is directly proportional to $b/R$, because more trajectories are mapped into the same bucket with the size of bucket increasing. It ensures that STN-Outlier finds enough evidences to classify one trajectory as inlier. By contrast, the F1-Measure decreases when $K$ enlarges. That is, with $K$ increasing, the trajectories in the same bucket must have a more similar encoding $(g(l_i))$, causing that one trajectory lacks enough R-near

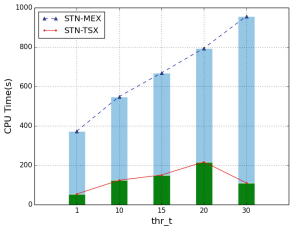



**Fig. 7.** The change of CPU time with varying $thr_t$

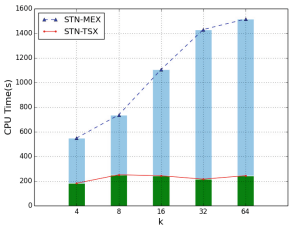



**Fig. 8.** The change of CPU time with varying $k$

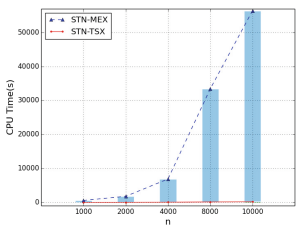



**Fig. 9.** The change of CPU time with varying $n$

trajectories to determine its status. However, we notice that when the buckets $(b/R)$ are enough big, the $K$ has less influence on F1-Measure and the results are near to that of the solutions without the TSX framework in Table. 1.

Figure 6 shows that the CPU time of algorithm is also directly proportional to $b/R$ due to one trajectory has to compare with more trajectories in one bucket to label its status. Then, the CPU time decreases with $K$ increasing for a fixed $b/R$. That is, an increase on collision probability caused by $K$ reduces the number of misclassification trajectories, and enables one trajectory not to compare with unnecessary trajectories.

In summary, the R-near subset could improve the efficiency of algorithm while ensure the effectiveness by selecting appropriate parameters.

**Varying Other Parameters.** Next we evaluate the efficiency of STN-Outlier using the real taxi data and synthetic datasets. We denote the MEX-based baseline solution for STN-Outlier as STN-MEX and the TSX-based solution as STN-TSX respectively. In particular, we fix the window size to 30, $d$ to 0.2, $\xi$ to 2,000 m and $R$ to 4,000 m for the experiments on the real taxi data, while fix the window size to 30, $d$ to 0.9, $\xi$ to 300 m and $R$ to 300 m for the synthetic datasets. In addition, the parameters $K$ and $b/R$ in LSH for all the cases are set to 10 and 100 respectively.

First, we evaluate the effect of varying the timebin count threshold $thr_t$ from 1 to the full window size. As shown in Fig. 7, STN-TSX are superior to the corresponding basic solution w.r.t the CPU time in all cases. Especially when $thr_t$ is set to the full window size, the STN-TSX outperforms the STN-MEX by a factor of 9x. We notice that the effect of STN-TXS decreases with $thr_t$ increasing. There are two reasons for this trend: (1) the STN-TSX only needs a few non-neighbor sub-trajectories (Time-aware Examination in Algorithm 1) to label the relationship of two trajectories as non-neighbor with $thr_t$ being big, and (2) the $R$-near subset of a trajectory is constant no matter how $thr_t$ varies.

Then, we vary the neighbor count threshold $k$ from 4 to 64. The results are shown as Fig. 8. The STN-TSX saves on average 76% of CPU time compared to the corresponding MEX solution. As the parameter $k$ increases, the CPU time of the STN-MEX increases linearly due to more neighbors have to be acquired to determine the status of a trajectory. By contrary, for the STN-TSX instead

we observe no sensitivity for varying $k$. This is because the STN-TSX finds the neighbors of $Tr_m$ only by searching its $R$-near subset that always remains unchanged no matter how large $k$ is.

Finally, we vary the number of trajectories $n$ from $1k$ to $10k$. In this case, we generate five synthetic datasets containing 1k to 10k trajectories. To eliminate the effect of variations in the outlier rates, we stabilize the outlier rate in all cases to around 4% by slightly adjusting the number of outliers. As expected, Fig. 9 shows that the CPU time cost of STN-Outlier increases linearly as the number of trajectories increases, since a trajectory must compare with more trajectories in the current window until it finds $k$ neighbors. Furthermore, it is obvious that STN-TSX exhibit much better performance than MEX-based solution. Especially, when $n$ is up to 10k, the factor can be more than 300x.

## 6 Conclusion

In this work we focus on the outlier detection on trajectory streams. After analyzing the requirements of trajectory stream applications, we introduce a distance-based trajectory outlier definitions. Considering the complex behaviors of trajectories over streams, we select sub-trajectory as the analytic unit and design a novel distance function. We introduce an optimized TSX framework scalable to big data trajectory streams. The experiments on real taxi data and synthetic datasets show that STN-Outlier can effectively and efficiently detect the abnormal objects over high-volume trajectory stream.

## References

1. Bu, Y., Chen, L., Fu, A. W.-C., Liu, D.: Efficient anomaly monitoring over moving object trajectory streams. In: Proceedings of SIGKDD 2009, pp. 159–168 (2009)
2. Aggarwal, C.C., et al.: A framework for clustering evolving data streams. In: International Conference on Very Large Data Bases 2003, pp. 81–92 (2003)
3. Knorr, E. M., Ng, R. T.: Algorithms for mining distance-based outliers in large datasets. In: Proceedings of VLDB 1998, pp. 392–403 (1998)
4. Lee, J.-G., Han, J., Li, X.: Trajectory outlier detection: a partition-and-detect framework. In: Proceedings of ICDE 2008, pp. 140–149 (2008)
5. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of SIGKDD 2003, pp. 226–235 (2003)
6. Yu, Y., Cao, L., Rundensteiner, E.A.: Detecting moving object outliers in massive-scale trajectory streams. In: Proceedings of SIGKDD 2014, pp. 422–431 (2014)
7. Li, X., Han, J., Kim, S.: Motion-alert: automatic anomaly detection in massive moving objects. In: Mehrotra, S., Zeng, D.D., Chen, H., Thuraisingham, B., Wang, F.-Y. (eds.) ISI 2006. LNCS, vol. 3975, pp. 166–177. Springer, Heidelberg (2006). https://doi.org/10.1007/11760146_15
8. Li, X., Han, J., et al.: Roam: rule- and motif-based anomaly detection in massive moving object data sets. In: SIAM International Conference on Data Mining (2007)
9. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: Proceedings of VLDB 2003, pp. 81–92 (2003)

10. Yuan, J., et al.: T-drive: enhancing driving directions with taxi drivers intelligence. J. Trans. Knowl. Data Eng. **25**(1), 220–23 (2013)
11. Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G., Huang, Y.: T-drive: driving directions based on taxi trajectories. In: GIS, pp. 99–108 (2010)
12. Freedman, D., Pisani, R., et al.: Statistics. W. W. Norton and Company, New York (2007)
13. Lee, J., Han, J., Whang, K.: Trajectory clustering: a partition-and-group framework. In: Proceedings of the ACM SIGMOD 2007, pp. 593–604. ACM (2007)
14. Galić, Z.: Spatio-Temporal Data Streams. SCS. Springer, New York (2016). https://doi.org/10.1007/978-1-4939-6575-5
15. Knorr E.M, Ng R.T.: A unified notion of outliers: properties and computation. In: Heckerman, D., Mannila, H., Pregibon, D., Uthurusamy, R. (eds) Proceedings of KDD, Newport Beach, CA, pp. 219–222. AAAI Press, Menlo Park (1997)
16. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: Proceedings of VLDB (1999)
17. Datar, M., Immorlica, N.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the Symposium on Computational Geometry (2004)
18. Wang, H., Su, H., et al.: An effectiveness study on trajectory similarity measures. In: Twenty-Fourth Australasian Database Conference, vol. 137, pp. 13–22 (2013)