Clustering-Based Trajectory Outlier Detection

Eman O. Eldawy[1]
Faculty of Computers and Information
Minia University, Minia, Egypt

Hoda M.O. Mokhtar[2]
Faculty of Computers and Artificial Intelligence
Cairo University, Cairo, Egypt

***Abstract*—The improvement in mobile computing techniques has generated massive trajectory data, which represent the mobility of moving objects like vehicles, animals, and people. Mining trajectory data and especially outlier detection in trajectory data is an attractive and challenging topic that fascinated many researchers. In this paper, we propose a Clustering-Based Trajectory Outlier Detection algorithm (CB-TOD). The proposed algorithm partitions a trajectory into line segments and decreases those line segments to a smaller set (Summary-trajectory SS(t)) without affecting the spatial properties of the original trajectory. After that the CB-TOD algorithm using a clustering method to detect the cluster with the smallest number of segments for a trajectory and a small number of neighbors to be sub-trajectory outliers for this trajectory. Also, our proposed algorithm can detect outlier trajectories in the dataset. The main advantage of CB-TOD algorithm is reducing the computational time for outlier detection especially for big trajectory data without affecting the efficiency of the outlier detection results. Experimental results demonstrate that CB-TOD outperforms the state of art existing algorithms in identifying outlier sub-trajectories and also outlier trajectories in real trajectory dataset.**

Keywords*—Data mining; outlier detection; trajectory data processing; clustering

I. Introduction

The various advances in GPS devices supported collecting an enormous number of moving objects data easily and rapidly. Therefore, mining of these trajectory data is insistently required to reveal and discover some unknown insights that could be employed to obtain intelligent transportation systems and facilitate smart cities' life. Generally, outlier detection in data mining relates to identifying an object that is incompatible with the other objects [1]. In mining of moving objects database, Trajectory Outlier Detection (TOD) is an important research topic. An outlier trajectory (anomalous) is a trajectory (or a segment of trajectory) that represent different characteristics than the majority trajectories in terms of similarity metrics [2-5]. Outlier segments in a trajectory are different segments from the other segments in the same trajectory as presented in [6], but the outlier trajectory is a trajectory having further few neighbors [4]. The identification of unusual trajectories has great importance in several applications. A popular application of detecting abnormal trajectories is the meteorological monitoring of typhoons. If we can identify unexpected variations in a typhoon path, like a variation in direction, we can announce an early warning for the reduction of casualties and property injuries as quickly as possible [7]. Also, identifying moving objects trends which may be events, represented by a group of animal moving objects in a specific time that does not conform to a familiar pattern, is essential for detecting animal abnormal habit and attracts the attention of biologists[6]. These applications are behind our motivation work presented in this paper. Outlier detection algorithms can be classified into four categories: distribution-based, distance-based, density-based and clustering-based [8].

Notwithstanding the value of trajectory outlier detection, especially detection sub-trajectory outliers, few research articles discussed this problem. Lee et al. [6] proposed a partition-and-detect framework (TRAOD) for detecting outlying sub-trajectories. TRAOD consists of two phases: partitions trajectories into segments, and then detects the outliers. In the partition phase, TRAOD separates each trajectory into a set of line segments. In the detection phase, density and distance-based measures employed to identify outlying sub-trajectories. Further, Zhang et al. [4] proposed the iBAT algorithm utilizing the isolation mechanism to distinguish outlier trajectories. Also, iBAT utilized a few in number and different than the majority as usual features of abnormal trajectories. However, the outlier trajectories recognized using the iBAT algorithm, but sub-trajectories outliers ignored.

Distinctive from static data, a trajectory may be long and has complicated characteristics. Hence, implementing the computations on the complete trajectory as a fundamental computational unit, it is presumably neglecting to detect local or global outlying partitions that may be essential for various applications.

Example 1: Suppose having five trajectories TR1, TR2, TR3, TR4, and TR5 as shown in Fig. 1. We observe that the thick part in Tr3 is an outlying sub-trajectory as it is different from the remaining partitions in the trajectory. Contrarily, if we compare the whole trajectory with its neighbors we can neglect these partitions because the deviations are averaged over the whole trajectory; so, the overall behavior of the trajectory TR3 appears to be similar to those of the neighboring trajectories.

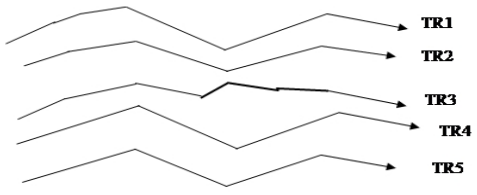


Fig. 1. Example of Sub-Trajectory Outlier.

Our proposed algorithm employs a partition-and-group framework for clustering trajectories [9] with some enhancements to reduce the computational cost. In our methodology, the coreset concept proposed in [10] used but without removing any partitions from trajectory. Basically, after partitioning the trajectories into a collection of line segments, these line segments decreased to a representative less set of lines without adjusting the length of the original trajectory (where the length of trajectory is the summation of the lengths of its line segments). After that, trajectories' partitions clustered employing a density-based spatial clustering of applications with noise (DBSCAN) clustering algorithm [11]. Density-based clustering methods proper for clustering a set of line segments as it identifies clusters of any random shapes. Furthermore, it operates efficiently in a big trajectory dataset [11]. Subsequently, the cluster with the fewest number of line segments for each trajectory in the dataset detected. If this cluster contains line segments that have inadequate neighbors, then the line segments of a trajectory in this cluster recognized outlier line segments for this trajectory. Moreover, if a trajectory contains a considerable number of outlying partitions, then identified it as an outlier trajectory.

In this paper, a Clustering-Based Trajectory Outlier Detection algorithm (CB-TOD) proposed. Our algorithm mainly consists of three phases:

1) Partitioning and summarization phase: each trajectory partitioned into several partitions (i.e. line segments); after that these partitions are reduced to a smaller representative set without affecting the information contained in the initial trajectory. Eventually, we get a summarized set of all partitions for all the trajectories in the dataset.

2) Clustering phase: similar line segments grouped to a cluster. Consequently, a cluster probably includes line segments from different trajectories.

3) Outlier detection phase: after clustering, for each trajectory, we get the cluster which includes the smallest number of segments for that trajectory and a small number of neighbors, then mark this cluster as an outlier cluster for this trajectory and accordingly classify the line segments included in this detected cluster as outlier segments. Moreover, we define an outlier trajectory as the trajectory with a considerable number of outlying partitions.

The main contributions in this paper are the following:

- We employed a novel model that reduces the computational time by decreasing the size of the trajectories dataset and representing each trajectory with the Summary set of line segments that are adequate to define the trajectory behavior without missing the basic motion information.
- A Clustering-Based Trajectory Outlier Detection algorithm (CB-TOD) proposed to detect outlier sub-trajectories as well as whole outlier trajectories utilizing a clustering-based methodology.
- Finally, experimental results are presented and demonstrate that CB-TOD outperforms existing algorithms in detecting both outlying sub-trajectories and outlier trajectories for real trajectory data. Also, the experiments confirm that CB-TOD reduces the computation time of outlier detection without affecting the accuracy of the outlier detection results.

The rest of the paper is structured as follows. Section II presents an overview of related work. Section III describes the problem statement. Our proposed clustering-based trajectory outlier detection (CB-TOD) algorithm presented in Section IV. Section V presents our experimental results. Section VI concludes the work presented in the paper. Finally, in Section VII, we suggest directions for future work.

II. Related Work

This section categorizes the previous research in trajectory outlier detection into two main directions: detecting sub-trajectories outliers and detecting outlier trajectories.

1) Sub-trajectories outlier detection: few research studies were conducted on the problem of detecting sub-trajectories outliers [6, 12-16]. TRAOD is the first approach for detecting outlying sub-trajectories[6]. TRAOD consists of two phases: firstly, partitions the trajectories and then detects the outliers. In the partition phase, TRAOD used the partition method used in TRACLUS algorithm[9]. Lee et al. [9] presented a TRACLUS algorithm that includes a partition-and-group framework for clustering trajectory data. TRACLUS consists of two steps: partitioning and grouping and used for clustering common sub-trajectories. In partitioning step, they applied the Minimum Description Length (MDL) principle[17] for partitioning a trajectory into a set of line segments. In the grouping step, they used a density-based clustering algorithm for grouping similar sub-trajectories. In the detection phase, TRAOD employed density and distance-based measures to detect outlying sub-trajectories. Despite the capability to detect outlying sub-trajectories and outlier trajectories, TRAOD suffered from computational time overhead as well as high complexity of $O(n^2)$. Later, Guan et al. [12] proposed R-Tree based Trajectory Outlier Detection (R-TRAOD) and used R-Tree to accelerate the process of outlier detection. Liu et al. [13] proposed a density-based trajectory outlier algorithm (DBTOD) and employed a density-based technique to detect outliers and solve the problems in TRAOD to detect outliers when a trajectory is local and dense. In[14] Daqing Zhang et al. proposed the iBOAT algorithm, which is an improvement on iBAT[4], to work in real-time data. Also, it determines which part(s) of a trajectory is an outlier. iBAT algorithm utilizes the isolation mechanism to identify the outlier trajectory. Despite, it can detect the outlier trajectories and neglect sub-trajectories outlier. In[15] Hao et al. proposed a probabilistic-model called DB-TOD, which models the drivers' behaviors from a historical trajectory dataset and assist in detecting outlier trajectories. DB-TOD used an automatic feature correction mechanism for modeling driving behaviors efficiently. Also, it can identify both complete

outlier trajectories and partial ones. Recently, Yu et al.[16] proposed a TODCSS algorithm that depends upon the common slices sub-sequence for identifying trajectory outlier. Firstly, they compute a direction-code sequence of each segment in each trajectory. Secondly, they used the common slices sub-sequences as a distance measure between two trajectories. Finally, the slice outliers and trajectory outliers discovered based on the new computed distance.

2) Trajectories outlier detection: many researchers studied mining in trajectories data to detect outlier trajectories [4, 18-20]. In [18] a framework called ROAM (Rule and Motif-based Anomaly Detection in Moving Objects) was presented. This framework introduces a motion-classifier for trajectory outlier detection. The motifs are a sequence of motion features with values related to time and location. The classifier distinguishes between an anomalous trajectory and a normal one. The main drawback on ROAM framework is that it requires labeled data for the classification process. Sabarish et al. [19] presented a trajectory Outlier Detection algorithm using Boundary (TODB). In TODB algorithm, they used the Convex hull algorithm to generate boundaries for trajectories. Furthermore, they exploit the ray casting algorithm as a classifier to judge a tested trajectory if its inside boundaries or not. The main drawbacks of TODB algorithm, because it used a classification method for categorizing trajectories, is that it required a labeled trajectories dataset that is rarely available. Also, it focuses on the whole trajectory and neglects the detection of sub-trajectories outliers. Moreover, Yong et al. [20] presented TOP-EYE algorithm that employed a decay function to identify the evolving trajectory in an advanced stage. TOP-EYE algorithm computes an outlying score for each trajectory in an accumulating method.

CB-TOD differentiates itself from previous studies by using clustering methodology to detect outlier sub-trajectories and also outlier trajectories. Moreover, the proposed CB-TOD approach decreases the computational time of detecting outliers by reducing the line segments comprising a trajectory and considering only the most representative segments.

III. Preliminaries

This section presents the preliminary concepts that will be used in the rest of the paper and formalizes the problem statement.

A. Definitions

Definition 1. A *line segment motion angle* θ is a representation of the segment's motion direction and it is measured as follows:

$$\theta = (\arctan\left(\frac{s.end_y - s.start_y}{s.end_x - s.start_x}\right)) * \left(\frac{180}{\pi}\right) \quad (1)$$

where the angle is defined by the two endpoints and the horizontal axis.

Definition 2. A *line segment* l is represented as (P_{start}, P_{end}, θ) where P_{start} is the start point of the segment, P_{end} is the end point of the segment, and θ is the motion angle of the segment and measured as in Equation 1.

Definition 3. A *Trajectory* τ is an ordered set of line segments, i.e. $\tau = \{l_1, l_2, l_3, \ldots l_m\}$, where m is the number of line segments in a trajectory τ.

Definition 4. Given a trajectory $\tau_i \in S$, a *Summary trajectory* of τ_i is a summarization representation of line segments in τ_i. Such that:

- If $|\tau_i| = m$, then $|SS(\tau_i)| = n$, such that $n \leq m$
- It mainly divides into two steps:

  a) Merge step:
  - if $(\theta_{i-1} - \theta_i) < \Phi_1$(accepted deviation angle) then
  - merge (l_i, l_{i-1}) into one-line segment $l_i' \in SS(\tau_i)$. Where $Len(l_i') = Len(l_i) + Len(l_{i-1})$

  b) Add without merge
  - if $(\theta_{i-1} - \theta_i) >= \Phi_2$(deviation angle)
  - Then $l_i \in SS(\tau_i)$

Definition 5. *Outlying line segments* of a trajectory τ_i called out (τ_i) is defined as following:

- Given a cluster C contains similar line segments depends on a distance measure.
- If C contains the minimum number of common line segments of this trajectory τ_i compared to other clusters (as the trajectory line segments may be divided among different clusters depends on the distance measure), and
- If C has a small number of similar neighbors' line segments from different trajectories in the dataset of trajectories S. In another words, if the number of participating trajectories in this cluster (we called it Density(C)) is less than a threshold P.

Definition 6. A trajectory τ_i is called *outlier trajectory* and added to outliers set if it contains a considerable length of outlying line segments. Such that:

$$\frac{the\ sum\ of\ the\ lengths\ of\ \mathrm{out}(\tau_i)}{the\ sum\ of\ the\ lengths\ of\ SS(\tau_i)} \geq F \quad (2)$$

where F is a threshold and its value depend on the length of a trajectory.

B. Problem Statement

Given a set of *trajectories* $S = \{\tau_1, \tau_2 \cdots \tau_n\}$, our goal is to detect the outlying line segments in each trajectory and also detect outliers' trajectories $Out = \{O_1, O_2 \cdots, O_{num}\}$ in a given dataset *S*. Our objective is minimizing the computation time of detection outliers by reducing the number of line segments in each trajectory to a representative once without losing the basic motion information of a trajectory.

IV. Clustering-Based Trajectory Outlier Detetcion (CB-TOD)

In this section, a description of the proposed approach Clustering-Based Trajectory Outlier Detection (CB-TOD) is presented. In CB-TOD we utilize the partition-and-group framework INTRODUCED IN [9]. Our approach is mainly divided into the following phases:

1) Trajectory partitioning and summarization phase
2) Clustering phase, and
3) Outlier detection phase

We explain these phases in the rest of this section. An overview of the proposed approach that abstracts the main steps in our algorithm is shown in Fig. 2. Also, Table I summarizes the main notations used in this paper.

A. Partitioning and Summarization Phase

This phase is a preprocessing phase for clustering. The input to this phase is the trajectories dataset *S*, then each trajectory in S is partitioned into a set of line segments by using the minimum description length (MDL) principle as presented in [9]. After that, a summary-trajectory set is created which is a summarization of a trajectory line segments. The coreset method is used for building the summary-trajectory set [10] with some modifications. In [10] the authors added to the coreset a segment with a high impact on the overall trajectory motion pattern and the segments with little effect in trajectory motion pattern are ignored; so, the trajectory-coreset is *a small* representative subset of the trajectory (that highly approximates the trajectory). In contrast, in our proposed approach a summary-trajectory includes segments that affected the motion pattern of a trajectory to a summary-trajectory set. Also, segments with a little effect on the trajectory motion pattern will be merged with the preceding segments to get a single segment with the total length of the merged segments and appended it to a summary-trajectory set.

A thresholds Φ_1 used for the allowance deviation angle and Φ_2 controls the deviation angle used in a summary-trajectory set. Thus, given two consecutive segments l_1, l_2 with a motion directions θ_1, θ_2 respectively as computed by Equation 1. If $(\theta_2 - \theta_1) \geq \Phi_2$ (deviation angle), then, l_2 is added to the summarized set, otherwise, if $(\theta_2 - \theta_1) < \Phi_1$ (accepted deviation angle), then we merge the two line segments (l_1, l_2) to get one-line segment (l_1'). Thus, we can consider the summary set as a representable set of the original trajectory whose total length is the same as the original trajectory length.

Example 2: A trajectory τ consists of the following line segments (l_1, l_2, l_3, l_4, l_5, l_6) as shown in Fig. 3, A segments l_1, l_2, and l_3 have the same motion direction and slope; so, we merge these segments into one-line segment and express it as l_1' and add it to the summary set of this trajectory. So, a summary-trajectory set will now consist of (l_1', l_4, l_5, l_6) line segments. The new set of line segments contains fewer segments which results in decreasing the comparison time for computing the distance between line segments. Furthermore, it does not affect the length of the resulting trajectory as shown in Fig. 4.

TABLE I. LIST OF NOTATIONS USED IN THIS PAPER

Symbol	Definition
S	Trajectory dataset
SS(τi)	Summary set of line segments for a trajectory τi
Len (l_i)	Length of line segment l_i
Φ_1	accepted deviation angle
Φ_2	Deviation angle
D	A set of line segments of all trajectories in trajectories dataset S
Density(C)	Number of participating trajectories in a cluster *C*
P	Threshold of acceptable number of participating trajectories in this cluster
F	Threshold for acceptable outlying partition in a trajectory

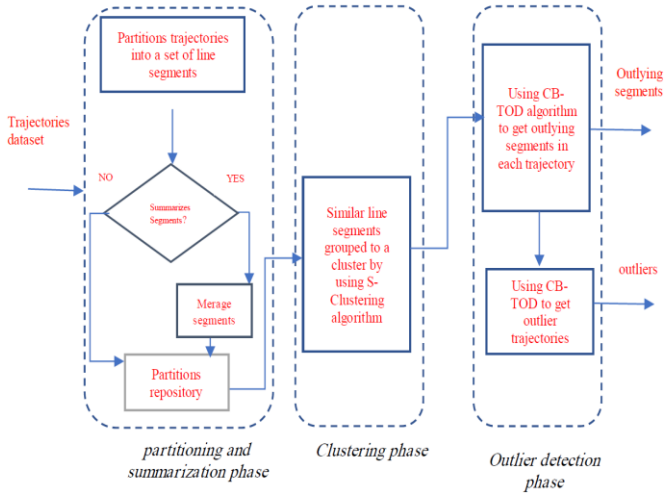


Fig. 2. Overview of CB-TOD.

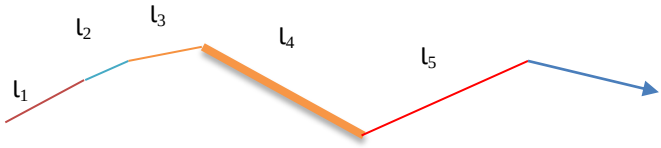


Fig. 3. Initial Trajectory Representation.

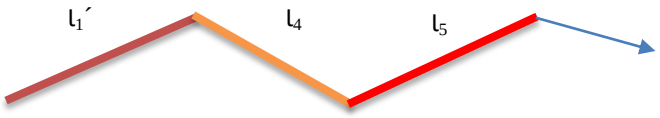


Fig. 4. Trajectory-Summary Example.

Algorithm 1 shows how to create the summary-trajectory set from the original trajectory. The input to the algorithm is the trajectory τ, the accepted deviation angle between segments Φ_1 and the deviation angle between segments Φ_2. The algorithm adds segments to the summary-trajectory ss*(τ)* if the absolute difference between its angular value and the preceding segment's angular value is greater than or equal to the deviation angle. Also, if the difference between the angular value of the current line segment and the angular value of the preceding line segment is less than Φ_1; then we extend the preceding line segment to be the result of merging the two segments (replace the end-point of preceding line

segment with the end-point of the current line segment) and then we add this line segment to the summary-trajectory ss*(τ)*.

A summary-trajectory algorithm is used for optimization and speed-up the computations of the distance between line segments.

Algorithm 1: Summary-trajectory (τ, Φ_1, Φ_2)

Input: List of segments in the given trajectory τ, Φ_1: the accepted deviation angle between segments' angular values, Φ_2 the deviation angle between segments
Output: *SS*(τ): List of summary segments in τ
1: $Seg_{Previous}$= P [0];
2: $Seg_{Current}$= P [1];
3: SS(τ). *Add* ($Seg_{Previous}$)
4. SS(τ). *Add* ($Seg_{Current}$)
5: **foreach** (*Seg* ∈ P(τ)) **do**
6: $Seg_{Current}$= *Seg;*
7: **if** ($|Seg_{Current}.angle - Seg_{Previous}.angle)| < \Phi_1$) **then**
8: *SS*(τ). *remove* ($SS(\tau)_{size-1}$)
9: *SS*(τ). *add* ($Seg_{Current}$)
10: **else**
11: **if** ($|Seg_{Current}.angle - Seg_{Previous}.angle) \geq \Phi_2$) **then**
12: *SS*(τ). Add ($Seg_{Current}$);
13: $Seg_{Previous}$=$Seg_{Current}$;
14: **end**
15: **output** *SS*(τ);

B. Clustering Phase

In our proposed approach a Density-Based clustering algorithm (DBSCAN) is applied to the summary-trajectory line segments set resulted from the previous phase. DBSCAN is a good choice for clustering large spatial databases [11] as it can discover any cluster with arbitrary shape. Moreover, using DBSCAN in clustering does not require knowing the number of clusters in advance. DBSCAN algorithm uses two parameters Ɛ and *MinPts* (where Ɛ is a parameter specifying the radius of a neighborhood concerning some point and *MinPts* is the minimum number of points required to form a dense region) [11]. In clustering, we used the same distance function as in [9]. Given a set D of line segments of all trajectories in the trajectory's dataset S. DBSCAN algorithm is then applied on D for grouping close line segments according to the distance. Notice that a cluster contains line segments from multiple trajectories to prevent constructing clusters with line segments from only one trajectory [9]. Algorithm 2 illustrates the pseudo code for S-Clustering (Summary-Clustering) algorithm and is used for clustering all line segments D in our trajectory's dataset S.

C. Outlier Detection Phase

In this phase, we get the set of clusters from the previous phase. Each cluster contains line segments that are close to each other. A cluster that includes the smallest number of segments for a trajectory and also has an insufficient number of neighbors is considered as an outlier cluster of this trajectory. Consequently, the line segments introduced in this detected cluster are classified as outlier segments. Moreover, the outlier trajectory is a trajectory that holds an observable length of outlying segments. Algorithm 3 describes a Clustering-Based Trajectory Outlier Detection Algorithm (CB-TOD). As demonstrated in algorithm 3, CB-TOD algorithm divides into two steps; firstly, we get outliers segments in each trajectory using clustering. Secondly, getting the outliers trajectories in the dataset by using outliers' segments. We sum the lengths of outlier segments of this trajectory and compared them to the total length of the trajectory as described in definition 6.

Algorithm 2: S-Clustering (summary clustering algorithm)

Input: A set of trajectories $S = \{ \tau_1, \tau_2 \cdots \tau_n \}$
Output: A set of clusters contains partitions segments for trajectory dataset
$C = \{C_1, C_2, \ldots\ldots\ldots\ldots, C_m\}$
1: **for each** ($\tau \in S$) do
2: /* Partitioning Phase*/
3: Summary-trajectory (τ, $\boldsymbol{\Phi}_1$, $\boldsymbol{\Phi}_2$)
/* Fig. 5 */
4: Get a set *SS*(τ) of line segments using the result;
5: Accumulate *SS*(τ) into a set *D*;
/* Grouping Phase */
6: Execute *Line Segment Clustering* on line segments in *D*;
7: Output a set *C* of clusters as the result;

Algorithm 3: Clustering-Based Trajectory Outlier Detection (CB-TOD)

Input: A set of trajectories $S = \{ \tau_1, \tau_2, \ldots \ldots, \tau_n \}$,
a set of clusters $C = \{C_1, C_2, \ldots\ldots\ldots\ldots, C_m\}$,
P acceptable number of participating trajectories in a cluster,
F threshold for acceptable outlying segments length of a trajectory.
Output: A set of outliers' trajectories $Out = \{O_1, O_2 \cdots, O_{num}\}$ with its outlying segments
1: **for each** ($\tau \in S$) **do**
2: **for each** ($C_i \in C$) **do**
3: /* Definition 5 */
4: min= C_1
5**: If** (min≥Count (ι (τ), C) && Density(C) ≥ P) **then**
6: min = Count (ι (τ), C)
7: Insert line segments on this cluster to out(τ)
8: **for each** $\tau \in S$ **do**
9: /* Definition 6 */
10: **if (**Len (*Out_seg* (τ)) ≥ *F)* **then**
11: insert τ into *Out*
12: Output *Out trajectories* with its outlying segment;

V. Experimental Evaluation

In this section, the performance of CB-TOD algorithm is evaluated experimentally.

A. Experimental Setting

CB-TOD algorithm is tested using the same animal movement data set as in [6,9,13] which represents Elk and Deer data. Elk data has 33 trajectories and 15,422 points; Deer data has 32 trajectories and 20,065 points. Our experiments are conducted on Intel core i7 2.7 GHz notebook with 8 GB of

main memory, running on the Windows 10 operating system. We implemented the algorithm using JAVA inside eclipse PHOTON IDE.

B. Accuracy Evaluation

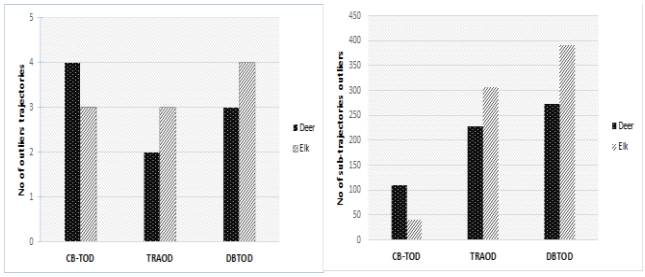
In this section, we evaluate the accuracy of our proposed algorithm CB-TOD. The accuracy measured by both the number of sub-trajectories outliers and trajectory outliers. In this experiment, we measure the number of anomalous trajectories and sub-trajectories for Elk data and Deer data as shown in Fig. 5 (a and b), respectively. We compare our obtained results with the results in [13], as we used the same datasets with the same parameter values. We observed that the CB-TOD algorithm detects fewer sub-trajectories outliers for both Elk and Deer data respectively, compared to TRAOD [6] and DBTOD [13] algorithms, as shown in Fig. 5(b). That is because we minimize the number of line segments in each trajectory by employing the summary-trajectory technique. Moreover, the CB-TOD algorithm discovers the same number of trajectory outliers compared to the TRAOD algorithm, as displayed in Fig. 5(a) for Elk data. Furthermore, in Fig. 5(a), we observe that our algorithm detects more numbers of trajectory outliers compared to TRAOD and DBTOD algorithms for Deer Data; that is because our algorithm decreases the representative trajectory line set without changing the information contained in the initial trajectory and that accomplished to us the accuracy goal.

Impact of deviation angle (Φ2). In this experiment, Fig. 6(a, b) displays the effects of varying the deviation angle (Φ2) on both the number of sub-trajectories outliers and the number of trajectories outliers. We evaluated the changes in the deviation angle (Φ2) and its effects on the number of outlier segments and the number of outliers trajectories in the dataset. Generally, when we increased the deviation angle (Φ2), the number of sub-trajectories reduced as it joined more numbers of segments that have the same motion. We observed that the best value for the deviation angle is between 60 and 120 degrees. A constant value for the accepted deviation angle Φ1 is used (Φ1≤30 degrees).

C. Performance Evaluation

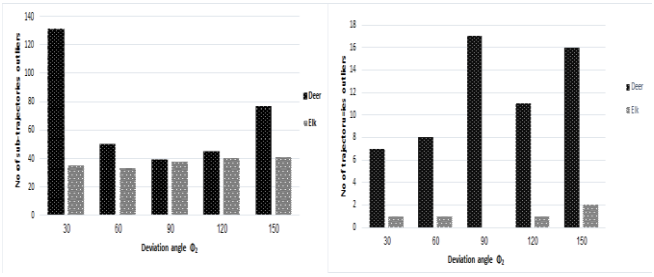
In this part of the experiments, we evaluate the run-time of the proposed algorithm (CB-TOD).

Computational time. Generally, the processing time of our proposed algorithm CB-TOD is less compared with the competitive outlier detection methods because of summarizing trajectory segments to a smaller set of segments without affecting the length of the original trajectory. We compared the processing time of our algorithm (CB-TOD) with both TRAOD [6] and DBTOD [13] algorithms, as we used the same datasets as in [13]. As shown in Fig. 7, the processing time of CB-TOD algorithm shows the best performance compared to both TRAOD and DBTOD algorithms for the two datasets (Elk and Deer), respectively. This is because using a summary-trajectories technique to reduce the computational time of the outlier algorithm leads to a reduction in dataset size (as it generates a fewer number of segments).



(a) Trajectories Outliers (b) Sub-trajectories Outliers

Fig. 5. Comparing between CB-TOD,TRAOD and DBTOD (Accuracy)



(a)Sub-trajectories Outliers (b) Trajectories Outliers

Fig. 6. Effects of Varying the Deviation Angle(Φ_2)

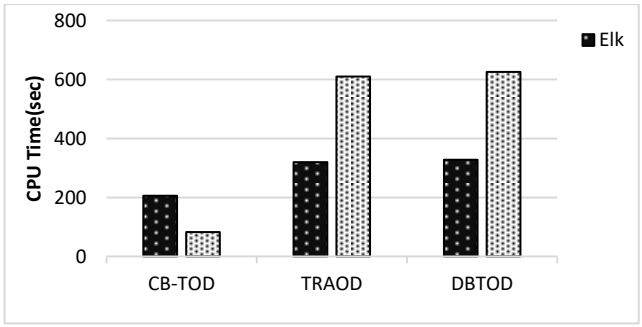


Fig. 7. Comparing between CB-TOD, TRAOD and DBTOD (Performance).

Impact of deviation angle (Φ2). In this experiment, the effect of varying the deviation angle (Φ2) on the processing time of CB-TOD algorithm is measured. As shown in Fig. 8, the processing time of CB-TOD decreased by increasing the value of the deviation angle (Φ2). The intuition behind this observation is that when we increase the deviation angle (Φ2); we get a smaller number of line segments and consequently the computation time decreases.

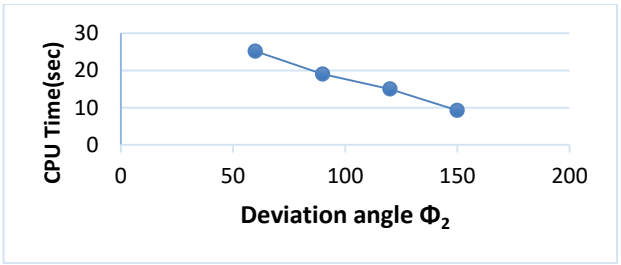


Fig. 8. Effects of Varying the Deviation Angle (Φ_2) on CB-TOD Running Time.

VI. Conclusion

In this paper, we proposed a clustering-based trajectory outlier detection (CB-TOD). Our algorithm summarizes the partitions of a trajectory to the smallest set of partitions without affecting the length of the original trajectory. CB-TOD can efficiently detect outlying sub-trajectory and also outlier trajectory from the trajectory dataset. The main advantage of CB-TOD algorithm is reducing the computational time of outlier detection especially for big trajectory data without affecting the efficiency of the outlier detection results.

VII. Future Work

For future work, we aimed to extend our work to maintain bigger datasets. Also, we will use machine learning techniques to predict possible outliers in a big trajectory dataset.

References

[1] W. I. D. J. M. K. Mining, "Data mining: Concepts and techniques," vol. 10, pp. 559-569, 2006.

[2] Y. J. A. T. o. I. S. Zheng and Technology, "Trajectory data mining: an overview," vol. 6, no. 3, pp. 1-41, 2015.

[3] L. Sun et al., "Real time anomalous trajectory detection and analysis," vol. 18, no. 3, pp. 341-356, 2013.

[4] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, and S. Li, "iBAT: detecting anomalous taxi trajectories from GPS traces," in Proceedings of the 13th international conference on Ubiquitous computing, 2011, pp. 99-108.

[5] J. D. Mazimpaka and S. Timpf, "Trajectory data mining: A review of methods and applications," Journal of Spatial Information Science, vol. 2016, no. 13, pp. 61-99, 2016.

[6] J.-G. Lee, J. Han, and X. Li, "Trajectory outlier detection: A partition-and-detect framework," in 2008 IEEE 24th International Conference on Data Engineering, 2008, pp. 140-149: IEEE.

[7] F. Meng, G. Yuan, S. Lv, Z. Wang, and S. Xia, "An overview on trajectory outlier detection," Artificial Intelligence Review, vol. 52, no. 4, pp. 2437-2456, 2019.

[8] D. Chen, C.-T. Lu, Y. Kou, and F. J. G. Chen, "On detecting spatial outliers," vol. 12, no. 4, pp. 455-475, 2008.

[9] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: a partition-and-group framework," in Proceedings of the 2007 ACM SIGMOD international conference on Management of data, 2007, pp. 593-604.

[10] O. Ossama, H. M. Mokhtar, and M. E. El-Sharkawi, "Clustering moving object trajectories using coresets," in Advances in Wireless, Mobile Networks and Applications: Springer, 2011, pp. 221-233.

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in Kdd, 1996, vol. 96, no. 34, pp. 226-231.

[12] B. Guan, Y. Zhang, L. Liu, J. Chen, and R. Guo, "An improving algorithm of trajectory outliers detection," in Advanced Technology in Teaching-Proceedings of the 2009 3rd International Conference on Teaching and Computational Science (WTCS 2009), 2012, pp. 907-914: Springer.

[13] Z. Liu, D. Pi, and J. Jiang, "Density-based trajectory outlier detection algorithm," Journal of Systems Engineering and Electronics, vol. 24, no. 2, pp. 335-340, 2013.

[14] C. Chen et al., "iBOAT: Isolation-based online anomalous trajectory detection," IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 2, pp. 806-818, 2013.

[15] H. Wu, W. Sun, and B. Zheng, "A fast trajectory outlier detection approach via driving behavior modeling," in Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 837-846.

[16] Q. Yu, Y. Luo, C. Chen, and X. Wang, "Trajectory outlier detection approach based on common slices sub-sequence," Applied Intelligence, vol. 48, no. 9, pp. 2661-2680, 2018.

[17] P. D. Grünwald, I. J. Myung, and M. A. Pitt, Advances in minimum description length: Theory and applications. MIT press, 2005.

[18] X. Li, J. Han, S. Kim, and H. Gonzalez, "Roam: Rule-and motif-based anomaly detection in massive moving object data sets," in Proceedings of the 2007 SIAM International Conference on Data Mining, 2007, pp. 273-284: SIAM.

[19] B. Sabarish, R. Karthi, and T. G. Kumar, "Spatial Outlier Detection Algorithm for Trajectory-Data," International Journal of Pure and Applied Mathematics, vol. 118, no. 7, pp. 325-331, 2018.

[20] Y. Ge, H. Xiong, Z.-h. Zhou, H. Ozdemir, J. Yu, and K. C. Lee, "Top-eye: Top-k evolving trajectory outlier detection," in Proceedings of the 19th ACM international conference on Information and knowledge management, 2010, pp. 1733-1736.