Unlicensed Taxi Detection Algorithm Based on Traffic Surveillance Data

Yao Tian*
Jiaming Yang*
21735083@zju.edu.cn
yjm@zju.edu.cn
Zhejiang University
Hangzhou, Zhejiang, China

Pengkai Lu
11721072@zju.edu.cn
Zhejiang University
Hangzhou, Zhejiang, China

ABSTRACT

Unlicensed taxi services severely disrupt traffic management and threaten passengers' safety. Traditional approaches to detect unlicensed taxis rely highly on manual work like collecting on-site evidence. However, these approaches are ineffective and inefficient. As to address this hardship, we propose an unlicensed taxi detection algorithm using pass-records data collected from surveillance cameras. First, based on spatio-temporal analysis, we propose path irregularity and time irregularity to distinguish commercial vehicles from non-commercial vehicles. Then, we evaluate the algorithm based on the actual vehicle pass-records data of Guiyang. The results show that our algorithm outperforms baselines in terms of accuracy and running time.

CCS CONCEPTS

• **Information systems** → *Spatial-temporal systems.*

KEYWORDS

unlicensed taxi, vehicle classification, time irregularity, path irregularity, urban computing

ACM Reference Format:
Yao Tian, Jiaming Yang, and Pengkai Lu. 2019. Unlicensed Taxi Detection Algorithm Based on Traffic Surveillance Data. In *5th ACM SIGSPATIAL International Workshop Emergency Management Using GIS (EMGIS'19), November 5, 2019, Chicago, IL, USA*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3356998.3365775

1 INTRODUCTION

Unlicensed taxis are taxicabs and other for-hire vehicles that are not duly licensed or permitted by the jurisdiction in which they operate. These vehicles pose a serious risk to passengers' safety and undermine traffic management. The traditional unlicensed taxi identification approaches taken by traffic management departments include collecting on-side evidence, watching videos etc., which require lots of manpower, material and financial resources. Fortunately, the rapid advances in video object identification technologies, especially license plate recognition technology, have given rise to new approaches to this problem. Different algorithms are developed with similar steps. Step 1: distinguish commercial vehicles from non-commercial vehicles, where commercial vehicles refer to taxi-like vehicles with the quite randomness and irregularity, such as taxis, unlicensed taxis, cars under the car-hailing scheme, etc.; non-commercial vehicles are otherwise, including private cars, official cars, etc.; step 2: exclude licensed taxis from commercial vehicles to get the unlicensed taxis. Step 2 is trivial in academic, because it only requires licensed taxis' labels from official departments, while step 1 is a challenging problem, for the difficulty in proposing distinguishable features under the premise of high efficiency. To the best of our knowledge, there is little relevant research on this problem and existing literature doesn't solve this problem well (details shall be provided in Section 2) and we try to assist the research of this problem by proposing an unlicensed taxi detection algorithm based on pass-records data collected from surveillance cameras.

The main work of this study is as follows:

1. We propose two notions, "path irregularity" and "time irregularity", as features for classifying vehicles. To further improve the classification effect, we propose a trajectory segment algorithm (TSA), which improves the discrimination of "path irregularity".
2. We build our algorithm using a real-world dataset in Guiyang, China, and compare the algorithm with two baselines in terms of accuracy and running time (see Section 6 for details). Experiment results demonstrate that our algorithm get more accurate classification results with less running time.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 presents the problem definition, basic concepts and dataset. The features are introduced qualitatively and defined quantitatively in Section 4. The specific algorithm steps are explained in Section 5. Experimental results are presented in Section 6. Finally, we conclude the paper in Section 7.

2 RELATED WORK

In this section, we first present related work on vehicle classification, and then we briefly introduce the shortest path computation which are used in our algorithm.

There has been extensive research in this area [6], which can be divided into several types according to the different types of data source such as vehicle classification based on GPS trajectory [2], vehicle classification based on social network check-in data [3],

*Both authors contributed equally to this research.

EMGIS'19, November 5, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-6965-7/19/11…$15.00
https://doi.org/10.1145/3356998.3365775

vehicle classification based on pass-records data [5][4]. We focus on the last one.

Yuan et al. [5] present an unlicensed taxi identification model (UTIM) composed of two sub-model components, candidate selection model (CSM) and candidate refined model (CRM). CSM is used to screen out coarse-grained suspected unlicensed taxis that are taken as an input for CRM to get a fine-grained list of suspected unlicensed taxis based on the support vector machine (SVM). However, the model in [5] has the following limitations: (1) The author only extracts classification features from perspective of passing records without considering the temporal and spatial characteristics of vehicles; (2) Some type labels are wrong: they treat high-end cars as non-commercial cars; (3) The model dimension (i.e. feature number) is too high, which results in low efficiency. By contrast, our classification model extract only two features from spatio-temporal perspective and adopt accurate commercial and non-commercial vehicles' labels.

Wang et al. [4]'s unlicensed taxis detection service (UTDS) tries to make improvements on the basis of [5], which considers the daily behavior features and sustainable behavior features of vehicles, and compares the classification effects of three kinds of machine learning methods: SVM, decision tree and logistic regression. However, the algorithm still has some imperfections: (1) Some features, such as "trip number", are not reasonable: they split the trajectory of a car into segments if the time interval of two consecutive records is longer than the threshold, without considering the matching performance of the distance with driving time and traffic congestion; (2) Time segmentation is unreasonable: they divide the working day time into two sections: 06:00-18:00 and 18:00-06:00 (the next day's 06:00), and extract the features from both sections. However, the behavior of vehicles varies greatly between 18:00-24:00 and 00:00-06:00, so these two sections should not be combined. (3) Feature redundancy: each day, they take the number of records and travel distance as a feature; however, this does not improve the classification effect and causes low computational efficiency. By contrast, we propose a different method to split the trajectory and choose features that are more discriminating, which guarantee not only the efficiency but also accuracy of our classification model.

Extensive research has been conducted on the shortest path computation. The common algorithms are Dijkstra's algorithm and A* algorithm. [1] is a good survey of the shortest path computation.

3 PROBLEM STATEMENT

In this section, we first give the problem definition and portray basic concepts, then briefly introduce the dataset in this paper.

3.1 Problem Definition

Given a vehicle set $S = \{c_1, c_2, \cdots, c_m\}$, design a binary classifier F to judge whether a vehicle is a commercial vehicle (CV) or a non-commercial vehicle (NCV), formally defined as:

$$F(c_i) = \begin{cases} 0 & (NCV) \\ 1 & (CV) \end{cases} .$$

Table 1: A SAMPLE OF PASS-RECORDS DATA

Timestamp	CarNo	Location	DeviceNo
2017-11-07 11:58:16	xxxx3S9	L1	5xxxx01
2017-11-07 01:21:44	xxxx3U2	L2	5xxxx04

Table 2: A SAMPLE OF SURVEILLANCE CAMERAS POSITION INFORMATION DATA

DeviceNo	Location	Longitude	Latitude
5xxxx15	L3	106.715300815	26.5643891339
5xxxx16	L4	106.646150216	26.6490947678

3.2 Basic Concepts

Definition 1. *(Record): A record is a tuple $p = p(c, x, t)$, where c represents a vehicle, x represents a surveillance camera, t represents time.*

Definition 2. *(Trajectory): A trajectory is a sequence of records in chronological order for a given vehicle c, e.g. $T_r : p(c, x_0, t_0) \rightarrow p(c, x_1, t_1) \rightarrow \cdots \rightarrow p(c, x_n, t_n)$.*

3.3 Dataset Description

The data used in this study includes four months of vehicles passing records from November 2017 to February 2018 in Guiyang, with approximate 800 million records, generated by more than 2,000 surveillance cameras, involving 1,071,358 vehicles. In addition, we use 500 known non-commercial vehicles and 6000 commercial vehicles to train the model. Some data samples are shown in Table 1, Table 2, Figure 1 (the red points in this fig represent surveillance cameras).

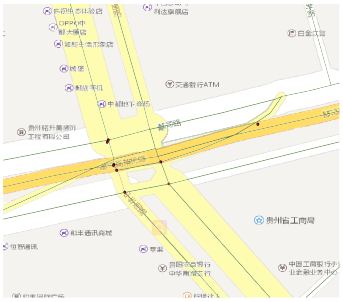

Figure 1: A SAMPLE OF GUIYANG DIGITAL MAP.

4 FEATURE ANALYSIS

In this section, we detail the process of proposing "path irregularity" and "time irregularity" as classification features.

4.1 Spatial Analysis

With increased travel demand and the popularity of car-hailing services, commercial vehicles have the following characteristics that distinguish them from non-commercial vehicles: the purpose

of the trip is diverse, the trips are independent of each other, and the interval between two trips is short. More specifically, for a given period of time, commercial vehicles may have more than one trip within this time period, but this is not the case for non-commercial vehicles. Take a taxi and a private car for example, within one hour such as 08:00-09:00, a taxi could probably finish many passengers' requests, while a private car may have only one trip during this time period. Consequently, for that taxi, the "real" driving distance is far greater than the shortest driving distance between the first passed surveillance camera and the last passed surveillance camera in this time period, while the above two distances are close for that private car. Note that shortest driving distance between two surveillance cameras here means the length of a path between two surveillance cameras on the digital map such that the sum of the lengths of road segments is minimized; "real" driving distance of a trajectory here means the sum of shortest driving distance between two consecutive record points of that trajectory. Admittedly, it is in practice unlikely for vehicles to always follow the strict shortest path between two consecutive record points, so it's just an approximation.

Therefore, we consider the ratio of "real" driving distance to the shortest driving distance between the first passed surveillance camera and the last passed surveillance camera of a trajectory to measure the difference. In general, this ratio of a commercial vehicle is greater than that of a non-commercial vehicle. On this basis, the idea of "path irregularity" is proposed. The following is the formal definition of "path irregularity":

Definition 3. *(Path Irregularity): Given a vehicle c, and a time period Δt, the trajectory of c in Δt is $T_r : p(c, x_0, t_0) \rightarrow p(c, x_1, t_1) \rightarrow \cdots \rightarrow p(c, x_n, t_n)$, let $T_r(k)$ be the k-th sub-trajectory divided by TSA, e.g. $T_r(k) = p(c, x_{k_0}, t_{k_0}) \rightarrow p(c, x_{k_0+1}, t_{k_0+1}) \rightarrow \cdots \rightarrow p(c, x_{k_1}, t_{k_1})$ $(0 \leq k_0 < k_1 \leq n)$. We say $l(k)$ is the k-th sub-path irregularity, where*

$$l(k) = \frac{\widehat{x_{k_0}x_{k_1}} - \overline{x_{k_0}x_{k_1}}}{\overline{x_{k_0}x_{k_1}}},$$

$$\widehat{x_{k_0}x_{k_1}} = \sum_{h=k_0}^{k_1-1} \overline{x_h x_{h+1}},$$

$\overline{x_i x_j}$ represents the shortest driving distance between x_i and x_j $i, j \in \{0, 1, \cdots, n\}$. The path irregularity of c in Δt is calculated as follows:

$$PathIr(c, \Delta t) = \sum_{k=0}^{M-1} \frac{|T_r(k)|}{|T_r|} l(k),$$

M represents the number of segments, $|T_r(k)|$ represents the number of records of $T_r(k)$, $|T_r|$ represents the number of records of T_r.

TSA refers to trajectory segmentation algorithm which is used before the calculation of "path irregularity", for it's crude to divide Δt equally into M segments and calculate "path irregularity" in each period of time. The same M is not suitable for all vehicles. TSA could decline the "path irregularity" of non-commercial vehicles by dividing the trajectory more rationally, and at the same time will not effectively divide the trajectory of commercial vehicles, that is, commercial vehicles still have more than one trip within the time period which causes high "path irregularity".

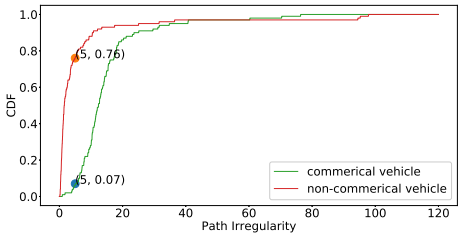

Figure 2: Path irregularity.

4.2 Temporal Analysis

By the knowledge and experience, the active time of non-commercial vehicles tends to concentrate on certain time periods, while the active time of commercial vehicles is more evenly distributed. For example, commuters are more likely to show up during the morning and evening rush hours; while commercial car divers might show up at any time. With reference to the definition of information entropy, we similarly give the definition of "time irregularity" to measure the uncertainty of vehicles' active time. The greater "time irregularity" is, the more uniform active time distribution is; the lower "time irregularity" is, the more concentrated active time distribution is. In other words, "time irregularity" of a commercial vehicle is generally greater than that of a non-commercial vehicle. Formal definition of "time irregularity" is as follows:

Definition 4. *(Time Irregularity): Given a vehicle c and a time period Δt, the trajectory of c in Δt is $T_r : p(c, x_0, t_0) \rightarrow p(c, x_1, t_1) \rightarrow \cdots \rightarrow p(c, x_n, t_n)$, we divide Δt into N disjoint segments equally, e.g. $\Delta t = \bigcup_{k=0}^{N-1} \Delta t_k$. Let*

$$P_k = \frac{\left|\left\{t_i \in \Delta t_k \middle| i = 0, 1, \cdots, n\right\}\right|}{n}.$$

The time irregularity of c in Δt is calculated as follows:

$$TimeIr(c, \Delta t) = -\sum_{k=0}^{N-1} P_k \lg(P_k).$$

In summary, we adopt "path irregularity", "time irregularity" as classification features.

We randomly select 100 non-commercial vehicles and 100 commercial vehicles, calculate their average daily "path irregularity" and "time irregularity" in their active days of a month, and plot the CDF, as shown in Figure 2, 3. We notice that 76% non-commercial vehicles' "path irregularities" are concentrated in 0-5, while only 7% commercial vehicles' "path irregularities" are concentrated in 0-5, and that 90% non-commercial vehicles' "time irregularities" are less than 0.7, while only 2% commercial vehicles' "time irregularities" are less than 0.7.

5 ALGORITHM

This section will detail the steps of our algorithms. The architecture is shown in Figure 4.

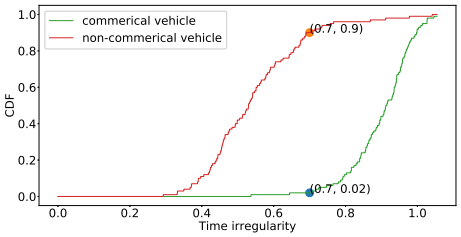

Figure 3: Time irregularity.

Table 3: A sample of data anomalies

Case	Exception type	Sample
1	the license plate number is garbled or vacant	Null, "A123"
2	Deviation of surveillance camera's GPS	<0.000000, 0.000000>

5.1 Preliminary

There are different levels of data anomalies in the vehicle passing records table, surveillance camera position information table and Guiyang City Digital Map. The common exceptions are shown in Table 3. Therefore, data preprocessing is necessary to improve the computational accuracy of subsequent algorithms.

For the case 1, we delete garbled or vacant records, and as for vehicles with very few records we set two thresholds: the number of active days per month (*ThreActiveDays*) and the number of records per day (*ThreNumRecords*), to filter out active vehicles.

For the case 2, specific processing steps are as follows:

Step 1: Use the ArcMap software to number the links (link ID) and nodes (node ID) in the digital map of Guiyang City, then generate the "RoadSection" layer.

Step 2: Use the surveillance camera position information table, combined with other information such as the road name and Google map, to adjust the latitude and longitude of more than 2000 surveillance cameras, then generate the "surveillance cameras" layer.

Step 3: Relate the "RoadSection" layer with the "surveillance cameras" layer, which corresponds to the surveillance camera number and the node ID

Step 4: Generate a "node ID-device No." correspondence Table

Step 5: Generate a weighted undirected graph G (V, E), where V is a set of vertices representing node IDs, and E is a set of edges representing the distance between a pair of vertices.

Step 6: Generate shortest path matrix between surveillance cameras using the Dijkstra algorithm.

5.2 Trajectory Segment Algorithm

Wang et al. [4] propose a trajectory segmentation method that splits the trajectory of a car into segments if the time interval of two consecutive records is longer than the threshold, and the author define "trip number" as the number of segments, and use it to classify vehicles. However, it can't handle the following common situations: (1) The time required to drive from one surveillance camera to another adjacent surveillance camera at "normal speed" may exceed the threshold; (2) Missed record can also cause the time interval between two consecutive records to be greater than the threshold; (3) Commercial vehicles rarely wait for passengers in a place for so long, so the statistics on the number of trips of commercial vehicles are inaccurate, which will result in no distinction of "trip number". We found the trajectory of a non-commercial vehicle can be divided into several segments by judging whether the distance between two consecutive records matches the driving time, however, for commercial vehicles they matched each other in most cases.

TSA we proposed takes into account factors such as the matching performance of the driving distance between two consecutive records and the travel time, as well as the travel time of all other vehicles passing through the same locations within similar time periods. See Algorithm 1 for details.

Algorithm 1 TSA

Input:

T_r: Trajectory needs to be split;
$ThreTime, ThreNumPairs$: Parameters as thresholds
$T(t)$: A function that takes time as input and output an integer as time_span(unit s)
$V(t)$: A function that takes time as input and output an integer as velocity(unit km/h)

Output:

A: The set of records which split T_r

$A = \emptyset$;
for $i = 0; i < length(T_r) - 1; i++$ **do**
  $\Delta t = T_r[i+1].t - T_r[i].t$;
  if $T_r[i+1].x == T_r[i].x$ **then**
    $A = A \cup \{T_r[i+1]\}$;
  else if $\Delta t <= ThreTime$ **then**
    continue;
  else
    $TimeList = FetchRefTime(carno, T_r[i].x, T_r[i+1].x, T_r[i].t, T(T_r[i].t))$; // See the appendix for details of $FetchRefTime$
    if $length(TimeList) < ThreNumPairs$ **then**
      $x_i = T_r[i].x; x_{i+1} = T_r[i+1].x; t_u = \frac{\overline{x_i x_{i+1}}}{V(T_r[i].t)}$;
    else
      $Q_1 = first\ quartile\ of\ TimeList$
      $Q_3 = third\ quartile\ of\ TimeList$
      $IQR = Q_3 - Q_1$
      $t_u = min(Q_3 + 1.5IQR, max(TimeList), \frac{\overline{x_i x_{i+1}}}{V(T_r[i].t)})$;
    if $\Delta t > t_u$ **then**
      $A = A \cup \{T_r[i+1]\}$;
return A

TSA successfully handles situation (1)(2). For situation (3), we use the feature of "path irregularity", rather than "trip number". In other words, we hope to divide the trajectory of non-commercial vehicles reasonably, thus lowering the "path irregularity"; meanwhile, we do not divide the trajectory of commercial vehicles strictly so as

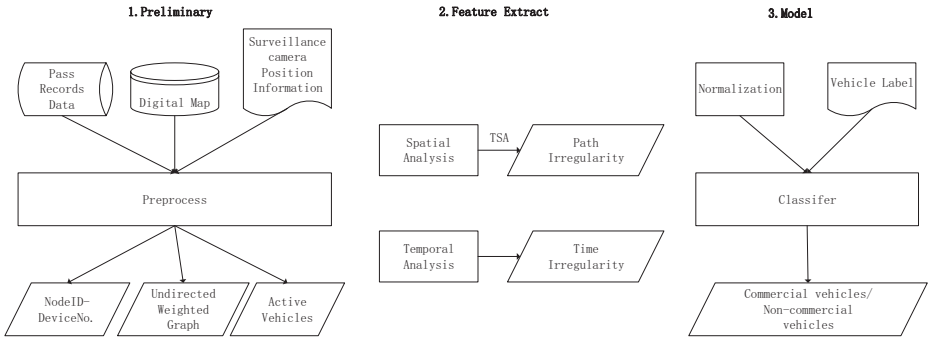

Figure 4: Architecture.

not to lower the "path irregularity". Therefore, situation (3) has also been successfully handled.

5.3 Feature Extraction

According to the feature definition mentioned in Section 4, we extract two features before training data. The feature extraction process is shown in Algorithm 2.

Algorithm 2 ExtractFeatures

Input:
c: vehicle;
TimePeriodSet: A set contains time periods;
Output:
$FeatureList$
$L_1 = [\ ], L_2 = [\ ]$; // initialize f_1, f_2 as empty lists
for each Δt in $TimePeriodSet$ **do**
  if c did not appear in Δt **then**
    continue
  else
    append PathIr(c,Δt) to L_1;
    append TimeIr(c,Δt) to L_2;
f_1=the z-score of L_1;
f_2=the z-score of L_2;
$FeatureList = [f_1, f_2]$

5.4 Unlicensed Taxi Detection Algorithm

Based on the extracted features, we use SVM to identify unlicensed taxis. Details of unlicensed taxi detection algorithm (UTDA) are shown in Algorithm 3.

Algorithm 3 UTDA

Input:
$VehicleSet$: Vehicles dataset;
$LabelSet$: $\{(c, l)\}$ c represents vehicle, l represents vehicle type;
$ThreActiveDays$: Threshold;
$TimePeriodSet$:The items in this set are time periods
Output:
$CVSet$: Commercial vehicle set;
$TrainSet = \emptyset$
for each (c, l) in $LabelSet$ **do**
  f_1, f_2=ExtractFeatures(c,TimePeriodSet);
  $TrainSet = TrainSet \cup \{(f_1, f_2, l)\}$;
Input $TrainSet$ into SVM model to train classifier F;
$CVSet = \emptyset$;
for each c in $VehicleSet$ **do**
  if $ActiveDays\ of\ c \geq ThreActiveDays$ **then**
    $label = F(ExtractFeatures(c))$
    if $label$==1 **then**
      $CVSet = CVSet \cup \{c\}$;

6 EXPERIMENTS

6.1 Experimental Settings

6.1.1 Computer settings.
We conduct the experiment on a PC with Windows 7 operating system, 32GB RAM and Intel Core I5-7300HQ 2.5GHZ CPU.

6.1.2 Parameter settings for UTDA.
ThreActiveDays = 9
ThreNumRecords = 3
TimePeriodSet = Weekdays in 2017-12
Kernal function for SVM : linear kernel
$N = 24$;
$ThreNumPairs = 10$;
$ThreTime = 10min$;

$$\begin{cases} T(t) = 3600, V(t) = 10km/h & if \quad t \in [00:00, 06:00] \\ T(t) = 1800, V(t) = 5km/h & if \quad t \in [06:00, 24:00] \end{cases}$$

6.1.3 Parameter settings for UTIM.
$ETBD\_heat = \{1, 2, 3\}$;
$morning\_rush\_hour = [07:00, 08:00]$;
$evening\_rush\_hour = [17:00, 18:00]$;
$feature\_cluster\_number = 10$;
$Gaussian\ kernel\ function\ parameter\ C = 8$;
$Gamma = 0.0078125$.
6.1.4 Parameter settings for UTDS.

Table 4: UTDA VS. UTDS VS. UTIM (ACCURACY ON TEST SET)

Size of Training Set	Accuracy on Test Set		
	UTDA	UTDS	UTIM
100	0.9400	0.7488	0.9099
200	0.9600	0.7614	0.9137
400	0.9600	0.7848	0.9060
600	0.9600	0.8190	0.9118
800	0.9680	0.8190	0.9137
1000	0.9700	0.8226	0.9118

$visited\ point\ threshold$: $30min$;
$Time\ Periods$: $[06:00, 18:00] \cup [18:00, 06:00]$;
$kernel\ function$: $linear\ kernel$.

6.2 Evaluation methods

We use accuracy, precision, recall, and F-score as metrics to evaluate the performance of our approach. FP is the number of non-commercial vehicles that are incorrectly classified as commercial vehicles; FN is the number of commercial vehicles that are incorrectly classified as non-commercial vehicles; TP is the number of commercial vehicles that are correctly classified; TN is the number of non-commercial vehicles that are correctly classified.

6.3 Accuracy comparison

In order to evaluate the effectiveness of our algorithm, we selected training sets of different sizes, such as training sets containing 100, 200, 400, 600, 800, and 1000 training samples, respectively, of which 40% are non-commercial vehicles, and 60% are commercial vehicles. The test set contains 200 non-commercial vehicles and 300 commercial vehicles.

We use UTIM and UTDS on our datasets and compare the accuracy of the three algorithms. As shown in Table 4, we use the same test set to evaluate the classification accuracy of different algorithms under the same training sets with size varying from 100 to 1000.

We can see that the accuracy of UTIM and UTDS is low when the number of training sets is small. When the number of training sets increases, the accuracy of UTIM and UTDS gradually increases, but there is still a gap with UTDA. Analysis of the results:

(1) The feature dimensions of UTIM and UTDS are much higher than the feature dimension of UTDA. When the number of training sets is insufficient, both UTIM and UTDS may have over-fitting problem, resulting in low accuracy.

(2) The features choosen by UTIM and UTDS are indistinguishable.

In addition, we also compare the precision, recall, and F-score of three models on different training sets. As shown in Table 5, Table 6, Table 7, we can see that UTDA has an advantage, regardless of the evaluation criteria.

6.4 Efficiency comparison

To evaluate the efficiency of our algorithm, we compared the model training time of our algorithm with UTDS and UTIM on different training sets. As shown in the Table 8, when the size of training set is small, Three models have similar training time, and UTIM's training time is slightly higher. However, when the size of training set increases, the training time of UTDS increases sharply, and the training time of UTIM also increases relatively slowly, while the maximum time of UTDA does not exceed 0.0130 s. From this, we can see that UTDA has an efficiency advantage on the premise of high accuracy.

Table 5: UTDA VS. UTDS VS. UTIM (PRECISION)

Size of Training Set	Precision on Test Set		
	UTDA	UTDS	UTIM
100	0.9599	0.7960	0.9047
200	0.9712	0.8216	0.9322
400	0.9724	0.8314	0.9158
600	0.9752	0.8392	0.8984
800	0.9848	0.8438	0.9054
1000	0.9775	0.8929	0.9630

Table 6: UTDA VS. UTDS VS. UTIM (RECALL)

Size of Training Set	Recall on Test Set		
	UTDA	UTDS	UTIM
100	0.9392	0.7817	0.9498
200	0.9619	0.7694	0.9233
400	0.9606	0.8045	0.9287
600	0.9577	0.8639	0.9618
800	0.9615	0.8570	0.9561
1000	0.9724	0.8003	0.8871

Table 7: UTDA VS. UTDS VS. UTIM (F-SCORE)

Size of Training Set	F-score on Test Set		
	UTDA	UTDS	UTIM
100	0.9495	0.7888	0.9267
200	0.9665	0.7946	0.9278
400	0.9665	0.8177	0.9222
600	0.9664	0.8514	0.9290
800	0.9730	0.8503	0.9301
1000	0.9749	0.8441	0.9235

Table 8: UTDA VS. UTDS VS. UTIM (MODEL TRAINING TIME)

Size of Training Set	Model training time (s)		
	UTDA	UTDS	UTIM
100	0.0079	0.2920	0.1610
200	0.0079	0.4190	0.3640
400	0.0089	0.5839	0.8400
600	0.0130	2.6900	1.5209
800	0.0119	7.8840	2.2731
1000	0.0120	13.1311	2.8527

7 CONCLUSTION

In this paper, we propose a new unlicensed taxi detection algorithm, which takes into account the differences in time and space between commercial and non-commercial vehicles, using only two features to identify commercial vehicles successfully. When tested with real data, the algorithm is not only superior in effectiveness to other algorithms but also greatly improved in efficiency.

REFERENCES

[1] Liping Fu, D Sun, and Laurence R Rilett. 2006. Heuristic shortest path algorithms for transportation applications: state of the art. *Computers & Operations Research* 33, 11 (2006), 3324–3343.

[2] Yong Ge, Hui Xiong, Chuanren Liu, and Zhi-Hua Zhou. 2011. A taxi driving fraud detection system. In *2011 IEEE 11th International Conference on Data Mining*. IEEE, 181–190.

[3] Luca Rossi and Mirco Musolesi. 2014. It's the way you check-in: identifying users in location-based social networks. In *Proceedings of the second ACM conference on Online social networks*. ACM, 215–226.

[4] Yujie Wang, Xiaoliang Fan, Xiao Liu, Chuanpan Zheng, Longbiao Chen, Cheng Wang, and Jonathan Li. 2017. Unlicensed taxis detection service based on large-scale vehicles mobility data. In *2017 IEEE International Conference on Web Services (ICWS)*. IEEE, 857–861.

[5] Wei Yuan, Pan Deng, Tarik Taleb, Jiafu Wan, and Chaofan Bi. 2016. An unlicensed taxi identification model based on big data analysis. *IEEE Transactions on Intelligent Transportation Systems* 17, 6 (2016), 1703–1713.

[6] Yu Zheng. 2015. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)* 6, 3 (2015), 29.

A FETCHREFTIME

We define FetchRefTime function to search for vehicles that passing through the same locations within similar time period and return their passing time

```
import datetime
import pyodbc
```

```
def FetchRefTime(carno: str, x1: str, x2: str, t:
     datetime.datetime, time_span: int):
  conn = "DSN=xxx;UID=xxx;PWD=xxx"
  cnxn = pyodbc.connect(conn)
  cursor = cnxn.cursor()
  time_str = t.strftime(r'%Y-%m-%d %H:%M:%S') # example
       2017-12-01 08:45:30
  traffic_table_name="xxx"
  query_str=''' select interval from (
                select carNo, ts, deviceNo, lead(ts) over
                     (partition by carNo order by ts)
                     nextts,
                lead(deviceNo) over (partition by carNo
                     order by ts) nextDeviceno,
                datediff(second, ts, nextts) interval from
                     {}) d
                where deviceNo = '{}' and nextDeviceno =
                     '{}' and abs(datediff(second, '{}',
                     ts)) < {} and carno !='{}'
                     '''.format( traffic_table_name , x1,
                     x2,time_str,time_span,carno)
  cursor.execute(query_str)
  result = [item[0] for item in cursor.fetchall()]
  cursor.close()
  cnxn.close()
  return result
```