UAS4T Competition - Team AtoZ - Aalto to Zhejiang

Roozbeh Mohammadi[1], Weiming Zhao[1], Qishen Zhou[2], Kaihang Zhang[2], Claudio Roncoli[1], Simon Hu[2],

I. Introduction

In this paper, we present a methodology developed for the extraction and estimation of traffic information from vehicle trajectories collected by drones in an experiment over the central district of Athens, Greece [1]. This is prepared to participate in the UAS4T Competition organised at the 23rd IEEE International Conference on Intelligent Transportation Systems (ITSC 2020). The proposed method is based on a combination of traffic dynamics analyses and advanced machine learning tools, which allows to extract all the required information. Notably, to obtain results, we utilise only the provided dataset, without using any external data, such as geographic information. Moreover, the proposed method is not tailored to the specific road sections, and it's transferable and applicable on any other similar dataset. As supplementary material to this article, we produced a Jupyter notebook (written in Python) that includes separate functions for each steps of the method, allowing readers to easily follow the workflow; in addition, numerical results are reported in a csv file. In the remainder of this paper we describe the main steps of proposed methodology. Further details, results, and figures can be found in the supplementary materials.

II. Methodology

In this section, we briefly describe the methods and algorithms used for the different components of our estimation method. To better help illustrate the concept, the whole methodology is summarised in Fig. 1.

A. Data pre-processing

In the pNEUMA dataset, every row represents the trajectory data of one vehicle, where, for each time step a vehicle is recorded, there are six columns of data; this results in having a structure with different number of columns for each vehicle. This structure is not convenient for utilising this dataset in efficient data analyses using, e.g., the Python package "pandas". Therefore, the first step is to reshape the dataset and convert it to a standard dataframe with a fixed number of columns. This way, each vehicle has multiple rows data in the reshaped dataset, i.e., one for each time step the vehicle is recorded.

The contest explicitly requires to analyse three areas separately, therefore we extract the corresponding data by defining polygons for each area of interest. Since each region may have multiple intersections, which happens for example in the red region, it turns useful to separate the data at the location of each intersection. Since the total stopping time decreases with respect to the increasing distance to the stop line, the dynamics of total stopping time is used to separate the data into multiple intersections. The region is divided into smaller segments with the same length along the road and the number of rows of stopping vehicle in each region is counted. Since the observation time step is the same across all rows, so the number of rows can represent the total number of stopping time in that segment. When the number of stopping rows decrease substantially from a high level, this is a good separation point between two intersections. After determining the separation points between intersections, the data for each intersection is extracted from the region data. Fig. 2 shows separated data for each segment, where the red region is separated into three segments. In the red region, there are also vehicles coming from the side roads and crossing the main road of interest. These data are out of our consideration, and they are removed.

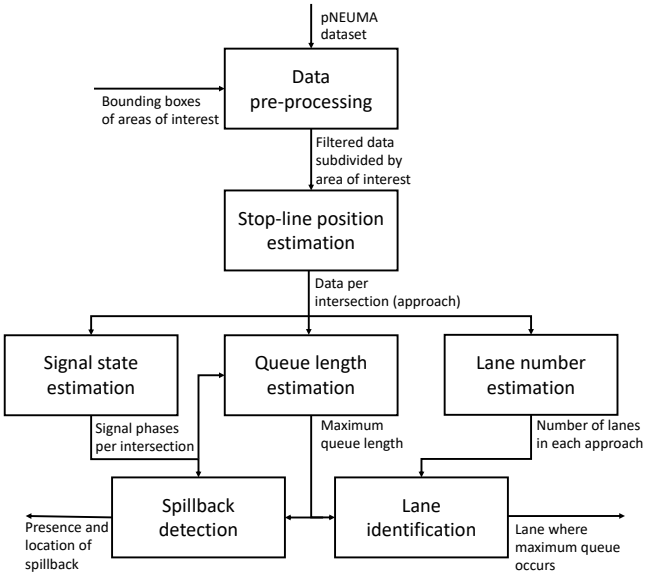

Fig. 1: Flowchart of the proposed methodology

B. Stop line position estimation

The stop-line position can be described as a line in the form $y = ax + b$, where the slope a and the intercept b need to be estimated. Since it is reasonable to assume that the stop-line is perpendicular to the street lanes, we start by finding the lane direction from vehicles trajectory data and then we obtain the slope a by considering a right angle.

[1]R. Mohammadi, W. Zhao, and C. Roncoli are with Aalto University, Finland, (e-mail: roozbeh.mohammadi@aalto.fi; weiming.zhao@aalto.fi; claudio.roncoli@aalto.fi).

[2]Q. Zhou and S. Hu are with Zhejiang University, China, (e-mail: simonhu@zju.edu.cn; qishenzhou@intl.zju.edu.cn).

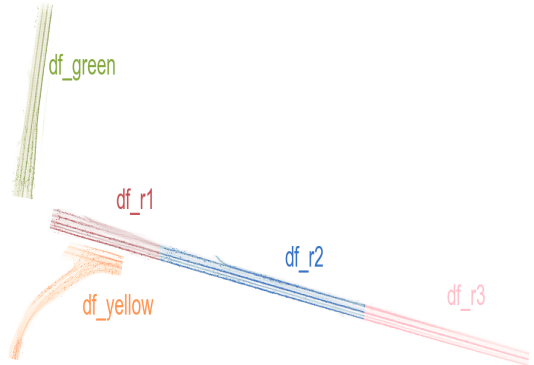

Fig. 2: Trajectories separated by areas (df_green, df_yellow, df_r*) and by intersections (df_r1, df_r2, df_r3)

In each cycle, vehicles line up behind the stop-line because of the red light, thus the position of the vehicle stopped at the head of the queue represents a reasonable approximation for the position of the stop-line. Considering that vehicles in a single lane in the queue are not allowed to overtake, the vehicles in the front queue theoretically start to stop first. Based on this, we select the vehicles which stop for more than a threshold (in this case, 10 s) from the multi-lane data, and extract the time these vehicles join and leave the queue, including them in a dataset named dataset-A. The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) method is used to cluster the join timestamps in dataset-A and find the data that belong to the same signal cycle.

Subsequently, we introduce dataset-B, which include the two earliest join timestamps for each class of dataset-A. Note that, as some vehicles do not comply with traffic rules, some stopped vehicles at the front of the queue may stop beyond the stop-line, while some conservative drivers may stop the vehicle in advance even though the vehicle in front has not stopped completely. In this case, the first vehicle to stop is not at the head of the queue and is far from the stop line. Therefore, it is not accurate to directly use the position information of these data to estimate the position of the stop-line. To solve the above problem, we use the location information in dataset-B to divide all the data in cells of 5 m (so that each cell contains at most one vehicle), find the interval with the most data, and then use the data location information in this interval to calculate the intercept b.

C. Signal state estimation

When vehicles form a queue during the red light of each traffic signal cycle, the time that vehicle at the head of the queue starts to move fluctuates around the green light start time. Consequently, we can use the dataset-A to estimate the green light starting time. Note that some vehicles at the head of queue will move slowly during the red period, and the start time of these vehicles is actually before the traffic light turns green.Therefore, it is necessary to remove the noise in this case. We choose again the machine learning method DBSCAN to cluster the leave timestamps in dataset-A. The DBSCAN method turns out to be very efficient and suitable to this problem in hand, as it does not need to input the number of clusters and can automatically remove the outliers. However, due to the limited number of data in some cycles, using the DBSCAN method for clustering may cause that this part of data to be judged as noise. In order to avoid this, firstly, we set a larger epochs and divide the data into different classes, which are in different cycles. Then, we apply the DBSCAN algorithm again for each class to remove noise. In this way, the earliest time stamp of each classes is the green light start time in each cycle. For verification of the results, we use the difference between the green light duration of adjacent cycles. The boxplot of the signal cycle in each intersection is shown in Fig. 3.

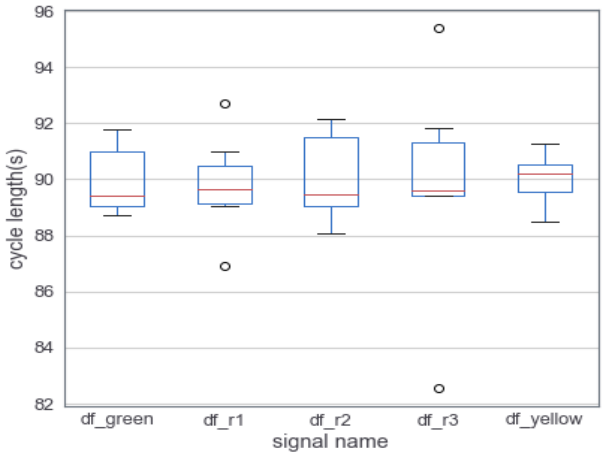

Fig. 3: Boxplot of results in the cycle length estimation

It can be seen that each intersection signal have a fixed cycle length with a stable period (of 90 s) and a small variance. This shows that the green light start time is reliable and effective. The same for the red time, we can use the join timestamp in dataset-A for estimation of vehicle coming to a stop event. The only difference is that we only need to use DNSCAN once this time. The timestamp of vehicles joining the queue will be scattered at the beginning of the red light, if we do twice DBSCAN like what we do in estimate green time, the scattered data will be judged as noise.

D. Maximum queue length estimation and spillback detection

According to the Highway Capacity Manual 2000 [2], a queue can be defined as "A line of vehicles, bicycles, or persons waiting to be served by the system in which the flow rate from the front of the queue determines the average speed within the queue. Slowly moving vehicles or people joining the rear of the queue are usually considered part of the queue. The internal queue dynamics can involve starts and stops. A faster-moving line of vehicles is often referred to as a moving queue or a platoon." Considering this definition, we assumed all stopped and slowly moving vehicles are included in queue by deploying a speed threshold to determine slowly moving vehicles. According to [2], a vehicle can be considered as queuing when it is moving with speed less than 3 m/s. A similar speed can be observed in Fig. 4, which shows a space time diagram of a sample from the dataset. Consequently,

we assume the same value as the threshold for filtering the queuing vehicles in this work.

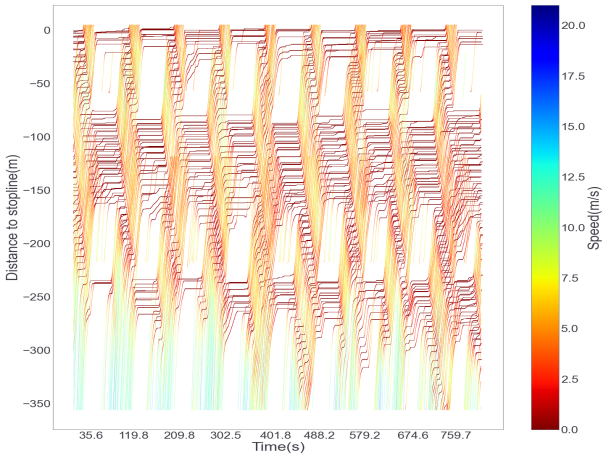

Fig. 4: Space time diagram of a sample data in the red area

We form a dataset of queuing vehicles by applying first the speed threshold and then, by assuming that, in the queued state, the distance between two adjacent vehicles in the queue is no more than 10 m. In order to avoid the phenomenon of abnormal vehicle spacing caused by overlapping of multiple periodic data on the road section, we need to divide the parking data according to the period. After dividing the cycle, study the queue lengths of different cycles: first, use the stop line to calculate the distance from the queuing data to the stop line. From the stop line to the downstream segment, cut the segment into multiple interval with 10 m long, and divide the data into each interval according to the distance from the vehicle to the stop line. Search each interval backward from the stop line to find the interval D with no data, then filter out the data between the stop line and the boundary line of the interval D. The maximum queuing length in this period is the maximum distance from the vehicle to the stop line after further filter. Note that we report as head and tail of a queue the coordinates of the vehicle, which correspond to its centre. If information on the vehicle length is available, it is feasible to consider this in our calculations and provide more accurate estimation of head and tail of the queue.

For the red area, the situation becomes more complicated. Although the red area is divided into three parts according to the intersection, in order to avoid the problem of queue length reduction caused by regional division errors, the paper combines the data of upstream and downstream segment to estimate the data of the upstream segment, which can not only reduce the queue length estimation error but also simultaneously identify the presence of spillback. Similarly, in order to further filter the data, we need to control the distance between the front and rear vehicles. In this time, when searching for the interval without data, we need to pay attention to the position of interval D which has no data in that. If the interval D completely locates in downstream segment, then mostly there is a spillback happend. In order to determine whether a spillback occurs, we need to compare the maximum queue length with the distance between the upstream and downstream segments of the stop line. If the maximum queue length is greater than the distance, then we need to locate the vehicles in the queue near the parking line of the downstream section and find out when and where the spillback occurred.

E. Lane detection

The divided segments in the process of data separation are used for lane detection because of the possible changes in number of lanes in the road and curving of roads. For each segment, the distance of each vehicle position to the boundary of the road is calculated. The motorcycle data are filtered out in the lane detection because motorcycles do not follow the lane rule closely and their data create noise in the lane detection. Then, the Gaussian Mixture Model(GMM), from the "scikit-learn" package, is applied to perform clustering, considering the position of both stopping and moving vehicles. The optimal number of clusters is determined by three factors: Akaike Information Criterion (AIC), difference of the mean of a cluster mean and number of points in the cluster. After the clustering, the lane index for each cluster is determined by the mean distance of the cluster to the road boundary. The leftmost lane along the road is labelled as lane 1, while the shoulder lane has the largest index. The lane information for every vehicle position is determined by predicting which cluster it belongs to. So when the position of maximum queue length is estimated, the lane that the maximum queue length belongs to can also be obtained. The graphical representation of lane identification of red area has been presented in Fig. 5.

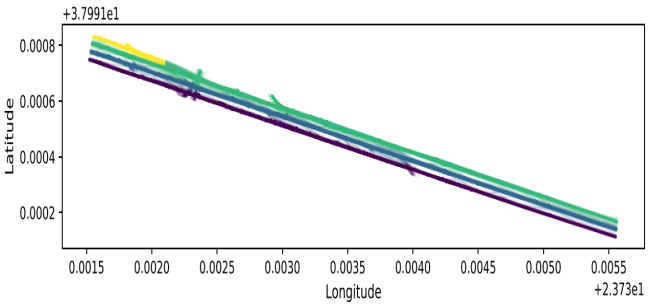

Fig. 5: Lane identification result in the red region

REFERENCES

[1] E. Barmpounakis and N. Geroliminis, "On the new era of urban traffic monitoring with massive drone data: The pneuma large-scale field experiment," *Transportation Research Part C: Emerging Technologies*, vol. 111, pp. 50 – 71, 2020.

[2] R. Dowling, "Traffic analysis toolbox volume vi: Definition, interpretation, and calculation of traffic analysis tools measures of effectiveness," Tech. Rep., 2007.