


Article

# Recognition of Intersection Traffic Regulations From Crowdsourced Data

Stefania Zourlidou <sup>1\*</sup> , Monika Sester <sup>1</sup>  and Shaohan Hu <sup>2</sup> 

<sup>1</sup> Institut für Kartographie und Geoinformatik, Leibniz Universität, Appelstraße 9a, 30167, Hannover, Germany; stefania.zourlidou@ikg.uni-hannover.de (S.Z.); monika.sester@ikg.uni-hannover.de (M.S.)

<sup>2</sup> Future Lab for Applied Research and Engineering (FLARE), JPMorgan Chase Bank, N.A., New York, NY; shaohan.hu@jpmchase.com (S.H.)

\* Correspondence: stefania.zourlidou@ikg.uni-hannover.de; Tel.: +49-5117-624-852 (S.Z.)

**Abstract:** In this paper, a new method is proposed to detect traffic regulations at intersections using GPS traces. The knowledge of traffic rules of regulated locations can help various location-based applications in the context of Smart Cities, such as the accurate estimation of travel time and fuel consumption from a starting point to a destination. Traffic regulations as map features, however, are surprisingly still largely absent from maps, although they do affect traffic flow which in turn affects vehicle idling time at intersections, fuel consumption, CO<sub>2</sub> emissions and arrival time. In addition, mapping them using surveying equipment is costly and any update process has severe time constraints. This fact is precisely the motivation for this study. Therefore, its objective is to propose an automatic, fast, scalable and inexpensive way to identify the type of intersection control (e.g. traffic lights, stop signs). A new method based on summarizing the collective behavior of vehicles crossing intersections is proposed. A modification of a well-known clustering algorithm for detecting stopping and decelerating events is presented. These detected events are then used to categorize vehicle crossing of intersections into four possible traffic categories (p1: free flow, p2: deceleration without stopping events, p3: only one stopping event, p4: more than one stopping event). The percentages of crossings of each class per junction arm, together with other speed/stop/deceleration features, extracted from trajectories, are then used as features to classify the junction arms according to their traffic control type (*dynamic* model). The classification results of the dynamic model are compared with those of the *static* model, where the classification features are extracted from OpenStreetMap. Finally, a *hybrid* model is also tested, where a combination of dynamic and static features is used, which outperforms the other two models. For each of the three models, two variants of the feature vector are tested: one where only features associated with a single junction arm are used (*one-arm model*) and another where features also from neighbouring junction arms of the same junction are used to classify an arm (*all-arm model*). The methodology was tested on three datasets and the results show that all-arm models perform better than single-arm models with an accuracy of 94% to 97%.

**Keywords:** traffic-rules; traffic-regulations; crowdsourcing; GPS-trace; trajectories; classification; movement patterns; clustering; collective-behaviour; smart city

## 1. Introduction

The idea of creating and sharing geographic information through individuals is not new. Individuals acting as sensors of their environment have been described as *citizens as sensors* [1], who can collect various kinds of data or share information from the environment they are in, such as photos, news, noise, speed measurements, air pollution data, etc. Since this data is combined with the geographical location from which it is taken, interesting information about these locations for a given time or a given period of time can be estimated, about a particular phenomenon, e.g. the noise level of a place, the speed limit of a road, etc.

The widespread use of modern mobile devices has opened up new possibilities for *spatial crowdsourcing* (SC), a term that describes "the potential of the crowd to perform real-

world tasks with strong spatial nature that are not supported by conventional crowdsourcing (CC) techniques" [2]. CC techniques lack the spatial element and focus on transactions conducted entirely over the Internet. In contrast, SC requires physical on-site presence and such information collected either opportunistically or participatively has increasing potential [3].

Some examples of leveraging data collected from individuals include automatic detection of road network changes and map updates using GPS trajectory data [4,5], pothole detection using crowd-sourced vehicle sensor data [6], estimation of road roughness from crowd-sourced bicycle acceleration measurements [7], and inferring the traffic state of roads by analysing the aggregated acoustic signal collected from the microphone sensor of the user's smartphone [8]. Predictions of phenomena such as earthquakes (earthquake early warning), which until recently required special equipment, can now be implemented using common consumer devices such as smartphones with low-cost sensors [9]. Another crowdsourcing-based service for citizens of large cities is information about the existence of vacant parking spaces near a destination [10]. Finally, crowdsourcing social media can also increase our understanding of human dynamics and spatio-temporal characteristics of cities and convey information about cities [11].

Other location-based applications that make our everyday life much easier are the accurate estimation of the travel time from a starting point to a destination, the elimination of false warnings in the advanced driver assistance systems offered by modern vehicles and the ambient sensing of autonomous vehicles, where traffic-related risks can be anticipated and driving actions can be planned accordingly. Furthermore, traffic regulators, such as traffic signals, affect significantly the traffic flow at intersections, which in turn influence the fuel consumption and air pollution. Intersections are one of the dominant locations where excessive fuel is consumed [12] and certain traffic regulations, i.e. traffic signals, contribute more to air pollution compared to others (e.g. stop signs), due to the excessive vehicle emissions that are observed at those regulated locations [13]. Therefore for environment-friendly and sustainable solutions related to daily commuting and traffic, such information is critical. Nevertheless, the type of localised traffic regulations, represented as map feature, surprisingly is still largely absent from National Agencies maps and from open maps such as OpenStreetMap (OSM) [14]. This study is motivated by this fact and the main research question addressed is how to automatically and cost-effectively identify traffic regulations using crowdsourced data.

Related studies, mainly use either GPS tracks or images [15]. Traffic sign recognition from in-vehicle cameras is a popular topic in the computer vision community, providing accurate detection of traffic signs [16,17]. However, although modern cars do have cameras, manufacturers do not share their data. But even if such data were available, their adoption the crowdsourcing scenario defined earlier has the disadvantage of generating a large amount of data (images) and therefore consuming resources such as bandwidth and storage space. Also, the cameras have to be placed in vehicles, adding further constraints for broad user participation. Another image-based approach could use images from street-level photos offered by platforms such as Google Street and Mapillary. As [Hu et al. \[18\]](#) point out, there are still many cities and places that are not covered by these services and therefore there are no images available to be crawled for traffic regulation detection. In contrast, GPS traces (i.e., time-ordered sequences of recorded locations) are compact representations of the successive locations that a moving object passes through over time and can be recorded without special equipment (e.g., mobile phone) and without the need to install a device on the front window of the car. For example, according to [19], an iPhone 6 has an overall average positional accuracy in an urban environment of 7-13 m (for different settings of seasons of year, times of day and WiFi usage period) and could therefore be used for this purpose. Therefore, we chose to use GPS traces to achieve our goal.

A systematic literature review of existing studies that use GPS traces for traffic regulation detection was conducted by [Zourlidou and Sester \[15\]](#). Here we review only some distinct studies of the field. [Saremi and Abdelzaher \[20\]](#) extract OSM features related to

speed as well as distance. In particular, they extract the speed rating of road segments, the distance of the nearest connected intersections, the end-to-end distance of the road to which an intersection belongs, the semi-distances of an intersection from both ends of the road to which it belongs, and the category of the road segment that characterizes its importance in the road network (e.g., primary, secondary, highway, etc.). In addition, on availability of dynamic crowd-sensed information (GPS tracks), the classification model incorporates features extracted from trajectories, such as traverse speed, number of stops and duration of the latest time interval the vehicle has stopped and been idling. A Random Forest classifier is trained to categorize three types of regulators: traffic signals, stop signs and unregulated intersections. The classification accuracy, with a confidence level of 80% in the prediction, is reported as 97%. Because a detailed classification report describing the results per regulator class is not provided, nor is a quantitative description of the datasets given (e.g., the number of regulators per regulator class), we cannot compare our results with this study, although we acknowledge this work as methodologically closest to our work.

Hu *et al.* [18] define two categories of classification features, *physical* and *statistical*. The physical features include the duration of the last stop before crossing the junction, the minimum crossing speed, the number of vehicle decelerations, the number of stops, and the distance of the last stop event from the intersection. Statistical features are defined as the minimum, maximum, average and variance of the physical features. The Random Forest classifier as well as Spectral Clustering were tested for a 3-category classification problem (traffic lights, stop signs and uncontrolled intersections), achieving accuracy above 90% for various feature settings and classification experiments.

Golze *et al.* [21] proposed a Random Forest classifier with oversampling and Bagging Booster to predict intersection regulators with 90.4% accuracy. Along with other physical features, such as the number of standstill events, the duration of standstill events, the mean distance from junction of all standstill events, the duration of the last standstill event, the distance from junction of the last standstill event, the mean speed and maximum speed while approaching the junction, they also calculate the percentage of trajectories with at least one standstill event. By also conducting a feature importance analysis, they show that the last feature is of great importance compared to other classification features.

Liao *et al.* [22] described a traffic light detection (binary classification problem) and impact assessing framework that can detect the presence of traffic signals and estimate the influence range of traffic lights (in space and time) using speed time series extracted from GPS trajectories and intersection-related features such as intersection type (connects arterial roads, connects secondary roads, connects arterial and secondary roads), road type (according to two speed limits) and traffic flow information. A distributed long short-term memory (DLSTM) neural network is used in the proposed framework, which treats discrete and sequential features separately and achieves an AUC value under the ROC curve of 0.95.

Méneroux *et al.* [23] detect traffic signals (binary classification problem) using speed profiles. By testing three different ways for feature extraction - functional analysis of speed recordings, raw speed measurements, and image recognition technique - they find that the functional description of speed profiles with wavelet transforms outperforms the other approaches. Random Forest classification achieved the best accuracy (95%) compared to the other tested classification techniques. However, the authors point out that the lack of data is a severe limitation of the experiments, as their dataset contained only 44 instances of traffic lights.

Last, Cheng *et al.* [24] propose a sequence-to-sequence framework for dealing with a three-class classification problem (traffic lights, priority signs and uncontrolled junctions) by feeding speed-profiles to a deep-learning classifier, showing that a Conditional Variational Autoencoder (CVAE) can predict regulators with 90% accuracy, outperforming the baseline model (a Random Forest classifier with 88% accuracy) that uses summarized statistics of movement as features.

This article addresses the traffic regulation recognition problem (TRR) by testing a new method in different datasets (three-category classification problem), under different trajectory and classification settings (number of trajectories, and one-arm features vs. all-arm features), which to our knowledge has not been done before. More specifically, the research contributions of the study presented in this article can be summarized as follows: it (1) proposes a modification of a well-known clustering technique for detecting short-term events such as stopping and slowing events (Section 2.2.1), (2) presents a new methodology for TRR by analysing GPS traces (Section 2.2.2), where in feature vector information from the adjacent junction arms is also included (all-arm models), (3) investigates the effect of turning trajectories on classification performance (Section 2.2.4.1), (4) examines the minimum number of trajectories per intersection required to achieve optimal accuracy (Section 2.2.4.2), (5) tests the methodology on three datasets including different groups of regulation types collected from different cities (Section 3) and (6) proposes an additional consistency check of the predicted labels at a junction level, correcting misclassified regulators when possible (Section 2.3). In the following section, we describe the datasets we used for the various experimentations as well as the proposed methodology.

## 2. Materials and Methods

### 2.1. Datasets

In this section we describe the data requirements for addressing the problem of traffic regulation recognition (TRR) from crowdsourced data (GPS trajectories and OSM), as well as the limitations under those requirements.

#### 2.1.1. Dataset requirements and limitations

As the proposed method is based on supervised classification, both GPS traces (for feature computation) and regulators (as labels) of intersection arms are needed. The process of labelling is time-consuming and poses general limitations for exploring the problem of detecting regulations, as such groundtruth map is always needed for training and/or validation purposes (more discussion on this limitation can be found at [15]). Although there are many open trajectory datasets that can be freely downloaded to use in the context of the research question we address here, the additional labelled data that are also required (the regulations of junctions) are not available.

Furthermore, another limitation regarding the trajectory dataset one can use is the sampling rate of GPS traces. Most open trajectory datasets have a low sampling rate (e.g., 1 sample every 15 seconds or per minute) and cannot be used to extract features such as stopping or deceleration events because between two GPS samples taken e.g., every 15 seconds, one or more stopping/deceleration events could occur and would not be detected. Therefore, having to deal with these challenges of the datasets, we were able to access three suitable datasets in total, which are described in the following section.

#### 2.1.2. Datasets for testing the proposed method

In Table 1 we give a description of the datasets we used to test the proposed method and to carry out experiments on the number of trajectories per intersection required for optimal classification accuracy (Section 2.2.4.1). The datasets contain various combinations of rules, with the Champaign [18] and Chicago [25] datasets containing the same rule classes (uncontrolled (UN), stop sign (SS) and traffic signals (TS)) and the Hanover dataset containing a subset of rules from the Champaign and Chicago datasets plus another regulator (UN, priority sign (PS) and TS). We consider one rule per intersection arm, which means that a three-way or a T-intersection has three rules and a four-way junction four rules. Hence, depending on the types of intersections (e.g. three-way, four-way, etc.), the total number of rules per dataset varies accordingly. The richest dataset in terms of rules is the Hanover dataset and the richest in terms of GPS traces (trajectories) is the Champaign dataset. Only, the Chicago trajectory dataset is publicly available [25]. The rest

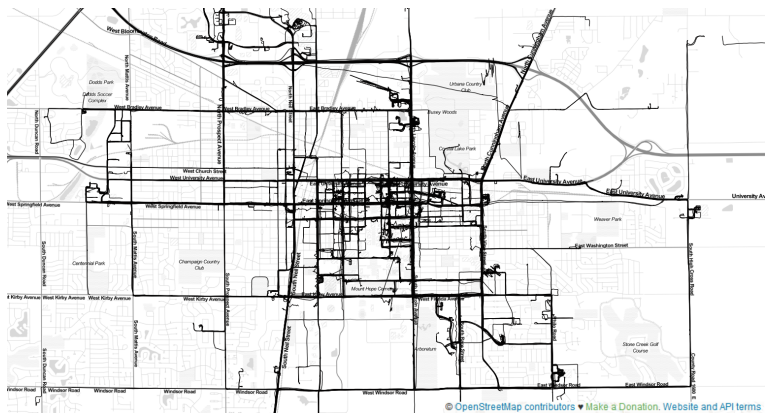


are self-collected. All data are naturalistic in the sense that drivers were not given external instructions on how to drive. Figure 1 illustrates the three datasets. 195

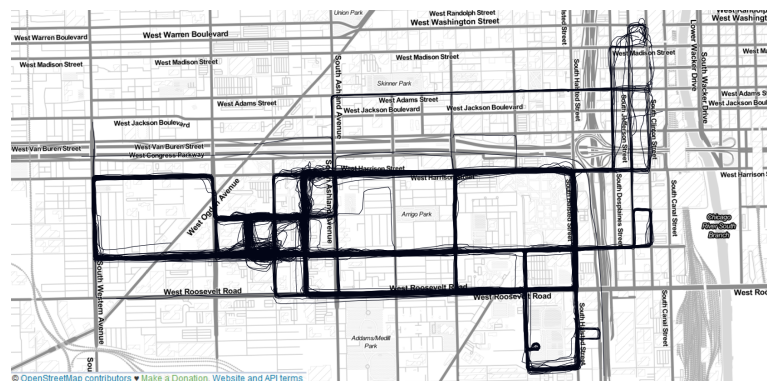
Furthermore, all but the Chicago groundtruth map of regulators were created manually by field observation (visiting all intersections and recording regulations). To obtain the regulations for the Chicago dataset, we used the Mapillary street imagery data [26], verified by other sources, and manually extracted the intersection rules. The Chicago groundtruth map is available here [27]. Moreover, all datasets except Chicago have a sampling rate of 1Hz on average, and Chicago has on average 0.28Hz. Finally, although the Hanover dataset contain Yield controlled arms (YS), most of them are sparsely sampled (few tracks cross them) and therefore only a few of them sampled from many tracks could in principle be used for training/testing. For this reason we excluded YS from our analysis. 196  
197  
198  
199  
200  
201  
202  
203  
204  
205



(a) Hanover dataset.



(b) Champaign dataset.



(c) Chicago dataset.

**Figure 1.** The three datasets that have been used in this study.

**Table 1.** Dataset used for testing the proposed methods.

City <sup>◊</sup>	Junc.	Rules	Traj.	Rules *
<i>Champaign (Illinois, US)</i>	713	2501	2202	TS, SS, UN
<i>Chicago (Illinois, US)</i>	156	568	889	TS, SS, UN
<i>Hanover (DE)</i>	1063	3538	1204	TS, PS, UN, (YS)

<sup>◊</sup> Country names: DE (Germany), US (United States); \* TS: Traffic Signals, SS: Stop Sign, UN: Uncontrolled, PS: Priority Sign, YS: Yield Sign (not used in the classification as the datasets do not contained enough YS controlled arms for training and testing).

## 2.2. Methodology 206

An important element of movement patterns are stop and deceleration events, which are detected based on a clustering approach. In this section we describe a modification of a known clustering algorithm for detecting short-term significant events (Section 2.2.1), as well as a new TRR methodology (Section 2.2.2) whose efficacy is examined under various settings (Section 2.2.4.1 and 2.2.4.2) 207  
208  
209  
210  
211

### 2.2.1. Clustering-based Stop and Deceleration Event Detection in Trajectories (the CB-SDoT algorithm) 212 213

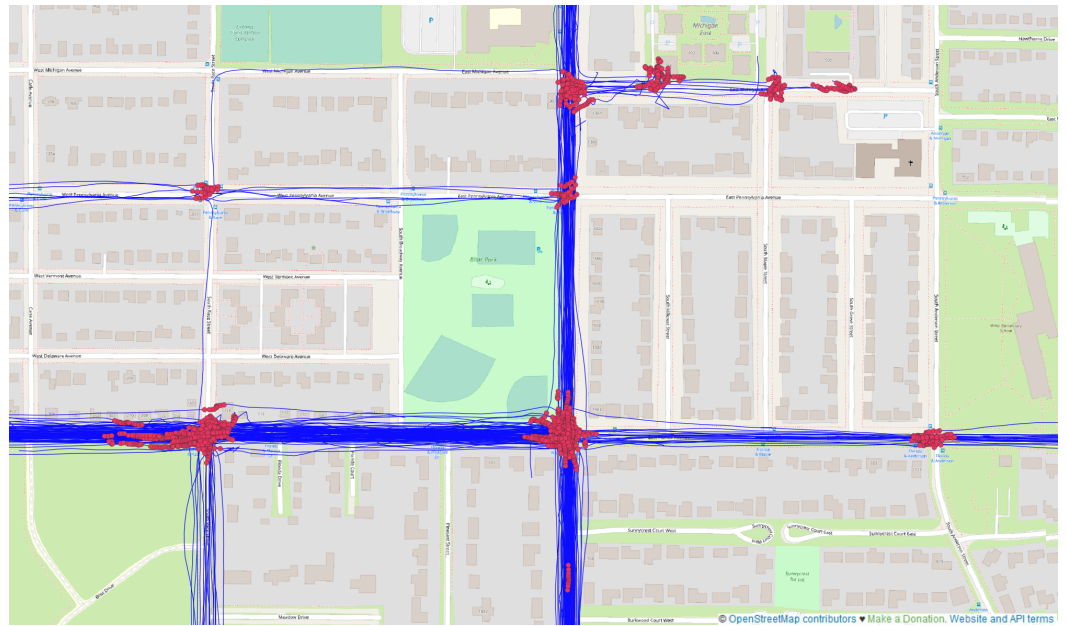
The CB-SDoT algorithm (see Algorithm 1) is a modification of the CB-SMoT algorithm [28] for detecting stop/deceleration events. CB-SMoT was originally proposed for discovering interesting places in trajectories. A place has the potential to be *interesting* [29] or *significant* [30], if someone spends a certain amount of time (i.e., over a time limit) in it. This means that by analyzing the trajectory of a moving object (e.g., a pedestrian, vehicle, animal, etc.), we can detect locations that are interesting to the observed object, given that it stayed there for a relatively long time. The problem of partitioning trajectories into sequences of *stops and moves* is a well-studied topic [31] and there are many different algorithms that provide solutions (e.g., [32], [33], [34]). 214  
215  
216  
217  
218  
219  
220  
221  
222

Here we adopted the CB-SMoT solution [28], which is a clustering technique that works similarly to the well-known density-based clustering algorithm DB-SCAN [35], but in addition to the distance between points, it takes into account the temporal distances between them to determine the clustering criteria. CB-SDoT identifies clusters of points (Figure 2) that within a certain distance  $Eps$  remain at least  $minTime$  and (unlike CB-SMoT) no more than  $maxTime$ . The extra  $maxTime$  restriction is required so that longer stops, not related to traffic events such as shop visits are not considered as interesting events. The values for these parameters were defined experimentally (stops:  $Eps=10$  m,  $minTime=4$  sec,  $maxTime=600$  sec, decelerations:  $Eps=10$  m,  $minTime=2.4$  sec,  $maxTime=3.9$  sec). Each detected cluster is a time-ordered sequence of points that represents a stopping or a deceleration event. For each cluster, we define a point as the *representative* of the cluster. We define such a point as the last point in the time series of the points of the detected cluster that has the lowest speed. The notions of *core point*, *linear neighborhood* and *neighboring points* refer to the same ones originally defined in [35] and [28] and for the sake of text space, we omit to give their definitions here. 223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237

### 2.2.2. Learning Traffic Regulators from Crowdsourced Data 238

#### 2.2.2.1 c-Dynamic Model: TRR via Summarization of Collective Movement Behaviour 239

The proposed regulation recognition method is based on the hypothesis that each regulator class *enforces* vehicles to move on certain moving patterns, and by detecting those patterns we can then *recover* the regulators. We describe the observed movement patterns by using as core elements (pattern blocks) the stop and deceleration events, as well as their non-observation, that is no stop and no deceleration event (four pattern blocks). For example, a movement pattern can be a free crossing of a junction where no stop or deceleration event is observed. Another pattern can be stopping only one time before crossing the junction. Numerous patterns can be defined by combining these patterns blocks. Then 240  
241  
242  
243  
244  
245  
246  
247



**Figure 2.** Stop events (red points) detected from the CB-SDoT algorithm in vehicle trajectories (blue lines).

**Data:**

$T$  : set of GPS trajectories

$Eps$  : interpoint distance

$minTime$  : minimum time

$maxTime$  : maximum time

**Result:** CB-SDoT identifies clusters of points that within a certain distance  $Eps$  remain at least  $minTime$  and no more than  $maxTime$ .

Returns: for each cluster with  $cluster\_id$ , the sequence of points of the cluster  $SeqPoints$ , the point representative  $RepCluster$  of the cluster and the duration  $Dur$  of the detected event

initialize  $clusters$  to an empty list

initialize all points of  $T$  as *unprocessed*

**for each trajectory  $t$  in  $T$  do**

**for each unprocessed point  $p$  in  $t$  do**

        // find the neighbors of  $p$

$neighbor\_list = linear\_neighborhood(p, Eps)$

**if  $p$  is a core point wrt  $Eps, minTime, maxTime$  then**

**for each neighbor  $n$  in  $neighbor\_list$  do**

$N\_neighbor\_list = linear\_neighborhood(n, Eps)$

$neighbor\_list = neighbor\_list \cup N\_neighbor\_list$

**end**

            add  $neighbor\_list$  as cluster with  $cluster\_id$  in  $clusters$

            find the  $RepCluster$  of the cluster compute the  $Dur$  of the cluster

            set all points in  $neighbor\_list$  as processed

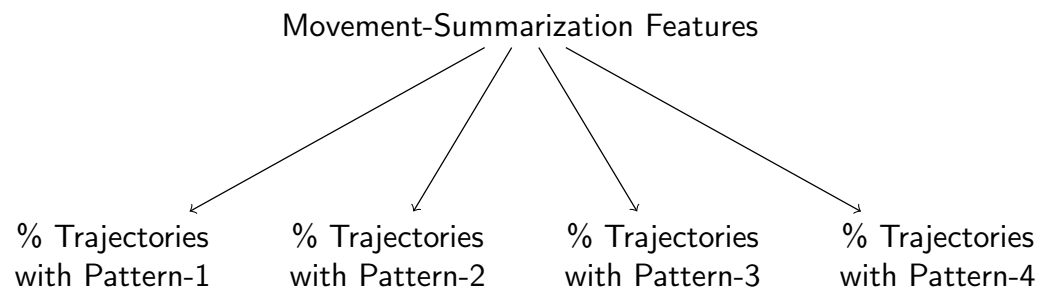
**end**

**end**

**end**

**Algorithm 1:** The CB-SDoT algorithm: Clustering-based Stop and Deceleration Event Detection in Trajectories

each regulated junction arm can be described from the movement patterns observed at its 248



**Pattern-1:** Free flow (no deceleration, no stop events)

**Pattern-2:** Deceleration with no stop events

**Pattern-3:** Only one stop event

**Pattern-4:** More than one stop event

**Figure 3.** The four movement patterns that describe a vehicle's crossing of a junction arm.

location, by simply summarizing the patterns (each described by stop/deceleration events) of all the trajectories that cross that junction arm.

For example, suppose  $N$  trajectories cross a junction arm  $i\_arm$ . From the  $N$  trajectories,  $M$  trajectories cross the  $i\_arm$  having a constant speed ( $p_1$ : free flow, i.e. no stop, no deceleration events) and  $N - M$  trajectories stop one time at the junction and wait for a few seconds ( $p_2$ : one stop before crossing the junction). We can then describe the  $i\_arm$  using the ratios of the trajectories following the two motion patterns,  $p_1$  and  $p_2$ . Defining  $p_1$  as the motion pattern of free flow and  $p_2$  as the motion pattern with stops, then  $i\_arm$  can be quantitatively described as a location where a *mixed* motion behavior is collectively observed and which can be summarized as follows:

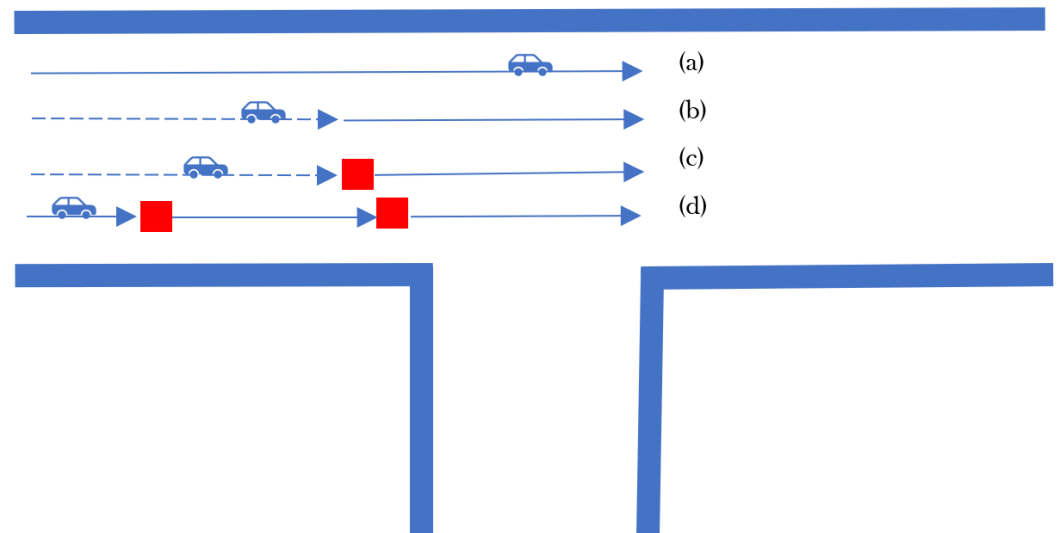
$$[p_1, p_2]_{i\_arm} = \left[ \frac{M}{N}, \frac{N - M}{N} \right], \text{ with } \sum_{n=1}^2 p_n = 1$$

Applying this idea in the context of our problem, we define four different movement patterns, depicted in Figure 3, instead of the two we used in the previous example ( $\sum_{n=1}^4 p_n = 1$ ):

- $p_1$ : Free-flowing (unobstructed) movement while crossing the intersection. Consequently, no deceleration or stopping events are observed.
- $p_2$ : The vehicle slows down without stopping.
- $p_3$ : The vehicle stops only once before crossing the intersection. However, it may slow down more than once.
- $p_4$ : The vehicle stops more than once before crossing the intersection.

Schematically, this idea is illustrated in Figure 4. Such a mixture of motion patterns has been used in [36], but in a different context. There, the goal was to dynamically determine the range of an intersection for obtaining the traffic flow speed and intersection delay under different traffic patterns. Here we define movement patterns for summarizing the collective behavior of vehicles at an intersection. The selection of the four patterns is intuitive and was motivated by the expected vehicle movement behavior at the intersections. For example, at a traffic light, we expect to observe a mixture of patterns overall, where proportionally patterns 1 and 4 are distinct compared to the corresponding values at a priority controlled intersection or at a priority sign. At a yield sign, we expect patterns 2 and 3 to have higher values compared to patterns 1 and 4. To confirm this intuition before applying the approach, we generated plots of vehicle speed profiles at various intersections, which were indeed consistent with our hypothesis.



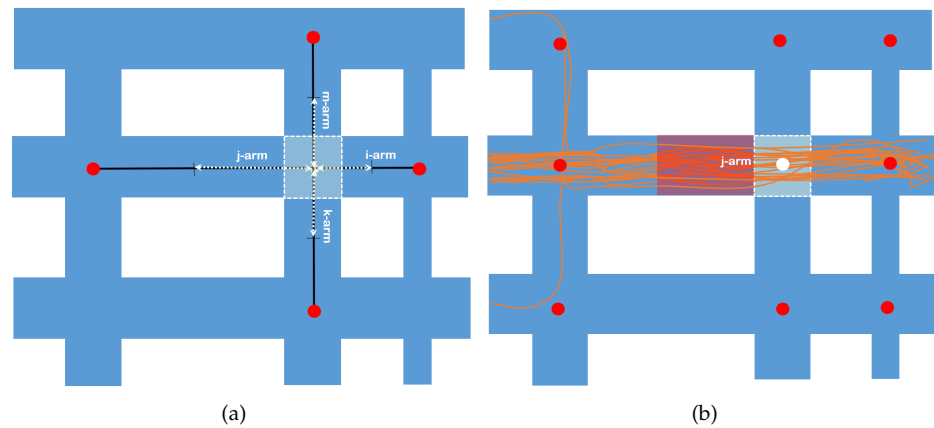


**Figure 4.** The four movement patterns that describe a vehicle's crossing of a junction: (a) unhindered crossing, (b) deceleration (dotted line) without stopping, (c) stop once (red dot), (d) stop more than once (here two stop events are depicted with the two red dots).

Along with these four patterns computed per junction arm, that are used as classification features, we add in the feature vector additionally the six percentiles of average speed (10th, 20th, 40th, 60th, 80th and 95th) of the trajectories that cross each junction arm. Therefore, a 10-dimensional feature vector (four pattern values plus six percentile values) is fed to the classifier for TRR. We refer to this method as *c*-dynamic model (*c*- stands for compact, we explain later the difference of the *c*-dynamic from the dynamic model). Figure 6 depicts the workflow of the proposed approach.

Regarding the implementation of this idea, all steps, from feature extraction to intersection classification, are expressed in the Algorithm 2. Since the problem is formulated as classification, we first extract the features that the classifier needs to learn how to map to the label space. As we explained earlier, each intersection is represented by ten features. We first compute all stop and deceleration events for each trajectory by using the CB-SDoT algorithm (Section 2.2.1).

Next, for each intersection arm of the dataset, we find all the trajectories that cross it, and for each trajectory we find the number of stopping and slowing events (if any) that occur within the half of the distance between the current intersection and the previous intersection visited, as well as the average speed. As Figure 5 depicts, for an intersection that has four arms,  $i\_arm$ ,  $m\_arm$ ,  $j\_arm$ ,  $k\_arm$ , for each arm and for each trajectory that crosses it, we compute the stop and deceleration events within the half distance that connects the intersection with the previously visited one (white arrows). According to the stop/deceleration events found in the trajectory that crosses an intersection arm, we categorize the movement behavior of each trajectory into one of four movement patterns. We then calculate the percentages of the trajectories for each pattern and for each intersection arm in the dataset.



**Figure 5.** Each intersection (red dots) is composed from intersection arms, that connect it to nearby intersections (black lines). Classification features are computed per arm, within half the distance of the road segment that connects the current arm with the previously visited one by the trajectory (white dotted arrows in (a)). For each trajectory (orange lines) in (b) that crosses the intersection arm  $j$ -arm from west to east, stop and deceleration events are computed within the distance indicated with the white arrow along  $j$ -arm in (a).

**Data:** GPS tracks, ground truth map (coordinates of junctions, traffic rules of junction arms)

**Result:** Label junction arms with the regulator type they are controlled with;

**while** not all trajectories have been processed **do**

    Find all stop events within the trajectory;

    Find all deceleration events in the trajectory;

    Add stop events in DB table *StopTB*;

    Add deceleration events in DB table *DecTB*;

**end**

**for**  $i \leftarrow 1, numJunctionarms$  **do**

    Find the trajectory  $TrjIds[]$  that cross the  $i$  junction arm ;

$numTrj \leftarrow$  number of  $TrjIds[]$ ;

**for**  $j \leftarrow 1, numTrj$  **do**

$Trj = TrjIds[j]$ ;

        Find the stop events of trajectory  $Trj$  that are along the 1/2 length of road segment between junction arm  $i$  and the previous visited junction arm ;

        Find the deceleration events of trajectory  $Trj$  that are along the 1/2 length of road segment between junction arm  $i$  and the previous visited junction arm ;

        Match the crossing behaviour (num. of stop/deceleration events) of the trajectory  $Trj$  to one of the four patterns;

        Estimate the average crossing speed

**end**

$p1, p2, p3, p4 \leftarrow$  Compute the % of the four patterns for junction arm  $i$  ;

$s1, s2, s3, s4, s5, s6 \leftarrow$  Compute the 10th, 20th, 40th, 60th, 80th and 95th average speed percentiles for junction arm  $i$  ;

    Add feature vector  $p1, p2, p3, p4, s1, s2, s3, s4, s5, s6$  to DB table *FeaturesTB* ;

**end**

Classification training and testing with data from *FeaturesTB* ;

Print classification report ;

**Algorithm 2:** Traffic regulation recognition from GPS tracks via summarization of movement patterns.

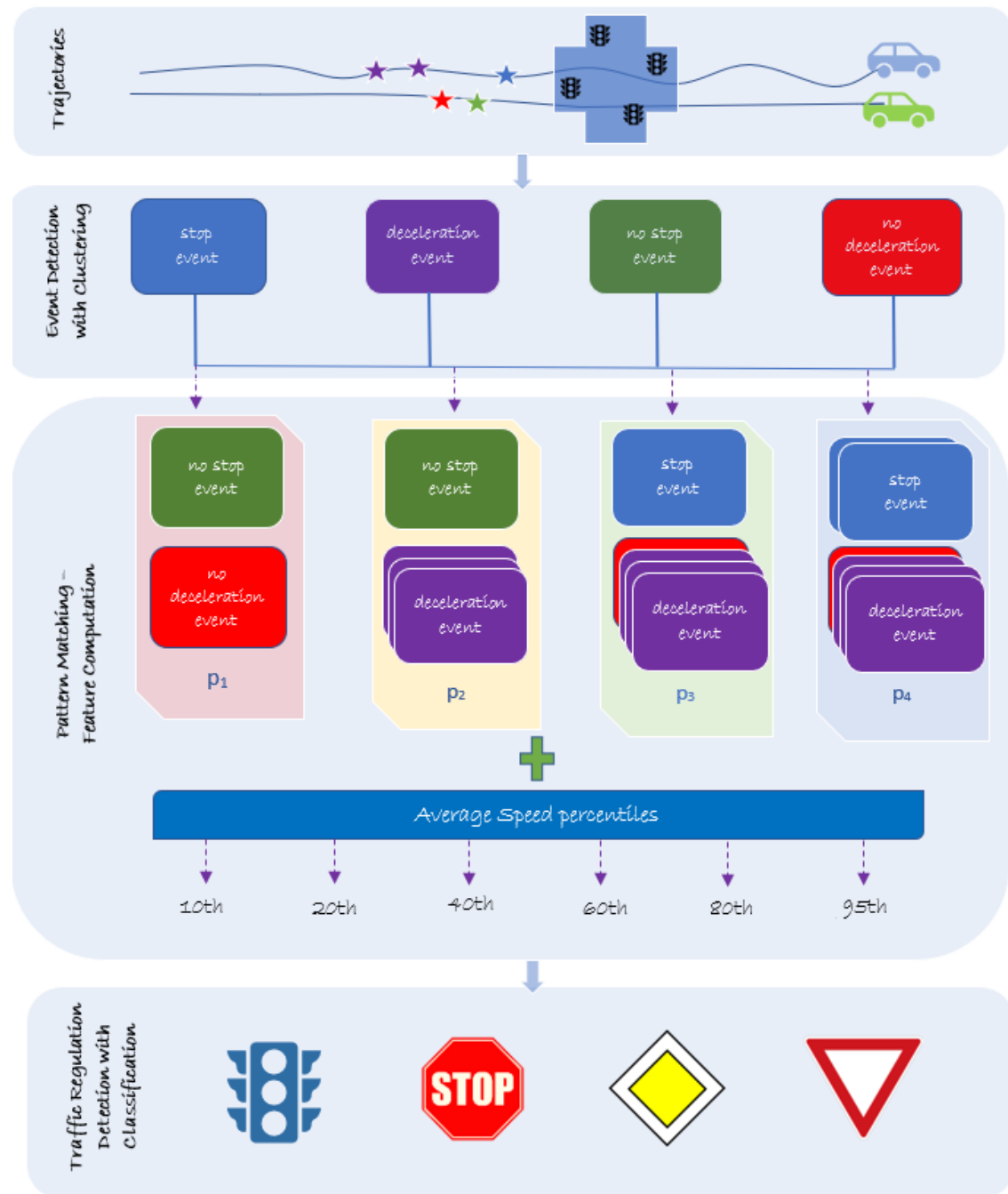


Figure 6. The steps of the proposed methodology for traffic regulation detection from GPS tracks.

## 2.2.2.2 Dynamic Model: TRR from GPS trajectories

The dynamic model can be considered as an extension of the c-dynamic model, using, in addition to the ten features used by the latter model, some statistical features (average, variance, minimum and maximum values) derived from the stopping and deceleration events and from the estimated vehicle speed. Compared to existing models, e.g., [18,20,21,37], this model has a richer feature vector (86 features in total) that includes more deceleration- and speed-related features. All features are enlisted in Table 2. The computation of the features and the steps of the TRR are done in a similar way as for the c-dynamic model.

**Table 2.** Overview of the features derived from the *dynamic* model.

#	Physical Feature*	Statistical Features **			
32	Number of stops	avg	var	min	max
	Duration of last stop	avg	var	min	max
	Duration of all stops	avg	var	min	max
	Mean Duration of all stops	avg	var	min	max
	Median Duration of all stops	avg	var	min	max
	Distance of last stop	avg	var	min	max
	Mean Distance of all stops	avg	var	min	max
	Median Distance of all stops	avg	var	min	max
32	Number of decel. events	avg	var	min	max
	Duration of last decel. event	avg	var	min	max
	Duration of all decel. events	avg	var	min	max
	Mean Duration of all decel. events	avg	var	min	max
	Median Duration of all decel. events	avg	var	min	max
	Distance of last decel. event	avg	var	min	max
	Mean Distance of all decel. events	avg	var	min	max
	Median Distance of all decel. events	avg	var	min	max
18	Minimum speed	avg	var	min	max
	Maximum speed	avg	var	min	max
	Average speed	avg	var	min	max
	Percentile avg speed (0.1)				
	Percentile avg speed (0.2)				
	Percentile avg speed (0.4)				
	Percentile avg speed (0.6)				
	Percentile avg speed (0.95)				
4	Traj. % with no stops/decels				
	Traj. % with decels				
	Traj. % with one stop				
	Traj. % with more than one stop				
Sum	86				

\* Derived per trajectory, \*\* Derived from the physical features per intersection arm, i.e. from all trajectories that cross the intersection approach. avg: average, var: variance, min: minimum, max: maximum.

### 2.2.2.3 Static Model: TRR from OSM Extracted Features

The static model uses features extracted from OSM, originally proposed from Saremi and Abdelzaher [20]. Each intersection approach (arm) is described by five features. The three of them regard street lengths and are illustrated in Figure 7. More specifically, the features of the static model are the following:

1. the **end-to-end distance** of the street that the intersection arm belongs to (light blue arrow in Figure 7). The length of a street is indicative of its importance in the street network. The same rationale applies also to the other distance-based features (2, 3).
2. the **semi-distance** of an intersection arm is the distance from the center of the junction to the center of the most distant intersection that the intersection arm is connected to (yellow arrow in Figure 7).
3. the **closest distance** of an intersection arm is the distance from the center of the junction that the arm belongs to, to the center of the nearest junction that the arm is connected to (green arrow in Figure 7).
4. the **maximum speed** of an intersection approach is the maximum allowed speed along it. Intersections controlled by a traffic signals in general have higher speed limit (e.g. 50 Km/h) compared to stop-sign controlled intersections (e.g. 30 km/h).
5. the **street category** refers to the street type category of the intersection arm (e.g. primary, secondary, tertiary, residential).



**Figure 7.** Illustration of the distance-related features of the static model along the north-south intersection approach of a four-way intersection (shown in red).

### 2.2.2.4 Hybrid Model: TRR from Crowdsourced Data (Dynamic and Static features)

The hybrid model uses the features from both the dynamic and static model, i.e. the 86 features of dynamic model and the 5 features of the static, in total 91 features.



### 2.2.3. One-Arm vs. All-Arm Models

So far, we have described four classification models (c-dynamic, dynamic, static, hybrid), where each intersection arm is represented by a set of features that is extracted solely from that arm (*one-arm models*). Motivated by the fact that for the classification of an intersection arm, information also from the adjacent intersection arms can be also relevant, we create for each model, the corresponding *all-arm model*, where each intersection arm is represented in the feature vector from a combination of features extracted from all the arms of the same junction.

For example the *i-arm* of the junction depicted in Figure 5(a) under the all-arm c-dynamic model has  $4(\text{arms}) \times 10(\text{features}) = 40$  features:

$$[p_1, p_2, p_3, p_4, s_1, s_2, s_3, s_4, s_5, s_6]_{i\_arm}, [p_1, p_2, p_3, p_4, s_1, s_2, s_3, s_4, s_5, s_6]_{k\_arm}, \\ [p_1, p_2, p_3, p_4, s_1, s_2, s_3, s_4, s_5, s_6]_{j\_arm}, [p_1, p_2, p_3, p_4, s_1, s_2, s_3, s_4, s_5, s_6]_{m\_arm}$$

The *j-arm* is similarly represented by the following feature vector:

$$[p_1, p_2, p_3, p_4, s_1, s_2, s_3, s_4, s_5, s_6]_{j\_arm}, [p_1, p_2, p_3, p_4, s_1, s_2, s_3, s_4, s_5, s_6]_{m\_arm}, \\ [p_1, p_2, p_3, p_4, s_1, s_2, s_3, s_4, s_5, s_6]_{i\_arm}, [p_1, p_2, p_3, p_4, s_1, s_2, s_3, s_4, s_5, s_6]_{k\_arm}$$

Only Saremi and Abdelzaher [20] have used such feature settings in their proposed static model, without having nevertheless tested the one-arm settings. From the existing trajectory based TRR methodologies, to our knowledge, none has examined such feature settings, therefore this study is the first to do so .

### 2.2.4. Testing Classification Performance Under Various Trajectory Settings

#### 2.2.4.1 The Effect of Turning/No-Turning Trajectories

Depending on the shape of the junction they are crossing, vehicles can go straight, turn left or right. This means that in the dataset we have straight or curved trajectories. The effect of turning at an intersection generally affects the driving behavior before and after the turn compared to a crossing by driving in a straight line, because the vehicle has to slow down before the turn and accelerate again after the turn. For this reason, other relevant studies have excluded curved trajectories from the dataset [18,21]. By excluding such trajectories, however, we reduce the data available to train and test the classifier. Therefore, we investigate the effect of using either all available trajectories (all combinations of right, left and straight trajectories) or exclusively straight trajectories on classification performance (Section 3.2).

#### 2.2.4.2 The Effect of Number of Trajectories

We examine the minimal number of trajectories per junction arm that are needed for applying the proposed method (Section 2.2.2). In addition, we investigate whether there is an optimal number of trajectories per intersection arm with which the classifier performs best. That is, how many trajectories do we need to have in order for the extracted patterns to be sufficiently *descriptive* for classification purposes? On the one hand, by setting a minimum number of trajectories as a condition in order for a junction arm to be selected for training/testing, we shrink the dataset: the higher this number, the fewer junction arms satisfy the condition, as most junction arms have only few trajectories. On the other hand, summarizing the collective movement behavior using only a few trajectories can lead to an incorrect representation of the *actual* behavior of the movement. We address this aspect of the problem by conducting experiments on the minimum number  $n$  trajectories that a junction arm must have to be included in the training-test: (a) using all trajectories that cross the junction arm and (b) using exactly  $n$  trajectories to compute the patterns of a junction arm.

Thus in (a), suppose we set the minimum number of trajectories to  $min = 10$ , we exclude from training and testing all junction arms crossed by less than 10 trajectories, and compute the patterns for the remained arms using *all* the trajectories that cross each

**Table 3.** Combinations of traffic regulators at intersections. In Champaign and Chicago there are UN/SS controlled junctions, all-way UN controlled junctions, all-way SS and all-way TS controlled junctions. In Hanover, there are UN/PS, YS/PS, all-way UN and all-way TS controlled junctions.

Dataset	three-way junctions							four-way junctions								
	UN SS	UN PS	YS PS	PS SS	UN (all)	SS (all)	TS (all)	Total	UN SS	UN PS	YS PS	PS SS	UN (all)	SS (all)	TS (all)	Total
Champaign	293	0	0	0	33	15	9	350	220	0	0	0	9	52	80	361
Chicago	36	0	0	0	4	8	8	56	10	0	0	0	0	17	71	98
Hanover	0	230	386	5	0	0	88	709	0	0	82	9	94	0	153	338

of them. If an intersection arm has 35 trajectories, we compute the patterns based on all 35 traces. In (b), conversely, having excluded crossing arms with less than 10 traces, we compute the patterns using *exactly* 10 trajectories (we selected the most recent ones).

### 2.3. Domain Knowledge Rules

The Table 3 shows the combinations of traffic regulators contained in each dataset. The Champaign dataset has 350 three-way junctions, 293 of which are controlled with UN/SS (UN-UN-SS), 33 are all-way UN, 15 are all-way SS, and 9 are all-way TS. Similar combinations are found in the Chicago dataset. In Hannover there are combinations of UN, PS, TS, SS and YS. The classification as explained previously is implemented at a junction-arm level, so after classification each junction has a predicted label for each arm sampled from trajectories. As Table 3 shows, the regulators are not randomly combined with each other, but there are underlying *domain knowledge rules* [20] that can be used in a post-classification step to *correct* misclassified arms by comparing the predicted arm labels at a junction level. Such simple knowledge rules were used in [20] and [18], however, without explaining how much they contribute to the overall accuracy. E.g., Saremi and Abdelzaher [20] use the following knowledge rule:

*Either all or none of the approaches contributed to the same intersection have a traffic light. This implies that when the classifier labels some of the approaches of an intersection, but not all of them, with TL, the predicted label should be revised. The revision makes either all or none of the intersection approaches have a traffic light, this being decided upon utilizing probabilities computed based on the fraction of decision trees voting for the approaches' alternative labels.*

Hu *et al.* [18] use another domain knowledge rule regarding T-junctions (denote A, B, and C for left, right and bottom ends of the junction):

*If A and B are UN, then C is SS controlled, else C has the same type as A and B.*

In defining the rules for domain knowledge, we also include the probability of the predictions, so that only predictions with high probability can be considered. We go one step further and compare the predictions of intersection arms belonging to the same intersection, both for correcting misclassified labels and for predicting labels for arms for which we have no data to make predictions. In the latter case, we make predictions for arms with missing data based on the predictions of the intersection arms of the same intersection for which we have data. E.g., For a three-way intersection, if one arm is predicted to be TS with high probability ( $>0.80$ ), we can infer that the other two (unlabeled) arms are also TS.

Another rule states that if in a three-way intersection there are two predictions for two arms of an intersection, one TS with probability 0.95 and the other SS with probability 0.79, we can conclude that the SS prediction is wrong, and therefore we correct SS to TS and also label the third unlabeled arm of the intersection as TS, so that the intersection conforms to the domain knowledge rule that *if one arm is TS, then all other arms of the intersection are TS*. We use 0.15 as the correction threshold, i.e., the difference between the two predictions, e.g., here  $0.95-0.79=0.16$ , so that the correction of the lowest predicted label is triggered.

**Table 4.** Classification accuracy of the TRR models.

	Method	Champaign		Chicago		Hanover	
		RF	GB	RF	GB	RF	GB
One-arm	c-Dynamic	0.93	0.93	0.78	0.77	0.86	0.85
	Dynamic	0.94	0.94	0.81	0.81	0.86	0.87
	Static	0.67	0.69	0.72	0.72	0.61	0.62
	Hybrid	0.94	0.95	0.82	0.82	0.87	0.88
All-arm	c-Dynamic	0.94	0.94	0.78	0.78	0.89	0.90
	Dynamic	0.94	0.95	0.83	0.84	0.90	0.91
	Static	0.86	0.86	0.89	<b>0.89</b>	0.86	0.87
	Hybrid-all static*	0.94	<b>0.95</b>	0.88	<b>0.91</b>	0.91	<b>0.95</b>
	Hybrid-all dynamic**	<b>0.95</b>	<b>0.95</b>	0.82	0.86	0.91	0.93
	Hybrid***	<b>0.95</b>	<b>0.95</b>	0.88	0.90	<b>0.92</b>	<b>0.95</b>

RF: Random Forest, GB: Gradient Boosting.

\* Only the dynamic features from one arm are included, along with the static features from all junction arms of the junction.

\*\* Only the static features from one arm are included, together with the dynamic features from all junction arms of the junction.

\*\*\* The model use the dynamic features from the adjacent junction arms and the static features of all the junction arms of the junction.

Similar consistency checks are performed for the other regulator combinations. Due to space constraints and the intuitive nature of the domain knowledge rules, we refrain from listing the other consistency checks.

#### 2.4. Classification Settings

Two tree-based classifiers are used for the classification of the intersection arms: the Random Forest and the Gradient Boosting classifier. For the implementation we used the XGBoost library [38], which has recently dominated many Kaggle competitions. All programming tasks have been implemented in Python 3.

As default model feature settings, we regard the features extracted from straight trajectories (we exclude the trajectories that turn at junctions). Moreover, junction arms that are crossed with less than five trajectories, are excluded from training and testing.

### 3. Results

In this section we list all the classification results of the experimentations discussed in the previous section (2.2). We first present the accuracy of the one-arm and all-arm models (Section 3.1). We then tune the best model and use it for all the other experiments regarding the effect of different trajectory settings on classification performance (Section 3.2 and 3.3).

#### 3.1. One-arm vs. All-arm Models

The Table 4 shows the classification accuracy of all models. We can see that the Gradient Boosting (GB) classifier performs as good or better than Random Forest (RF) for almost all experiments. Only in the c-Dynamic model for the Chicago and Hannover dataset, RF performs slightly better than GB (+0.1 accuracy).

Comparing the one-arm models with each other, the static model has much lower accuracy than the other models for all datasets. The hybrid model has the best accuracy (0.95 in Champaign and Hanover, and 0.82 in Chicago) and the dynamic model performs better than the c-dynamic but worst than the hybrid.

With respect to the all-arm models, we observe that the static model performs much better than the one-arm model, but only for the Chicago dataset does it manage to outperform the c-dynamic. In all other experiments the other models have better accuracy.

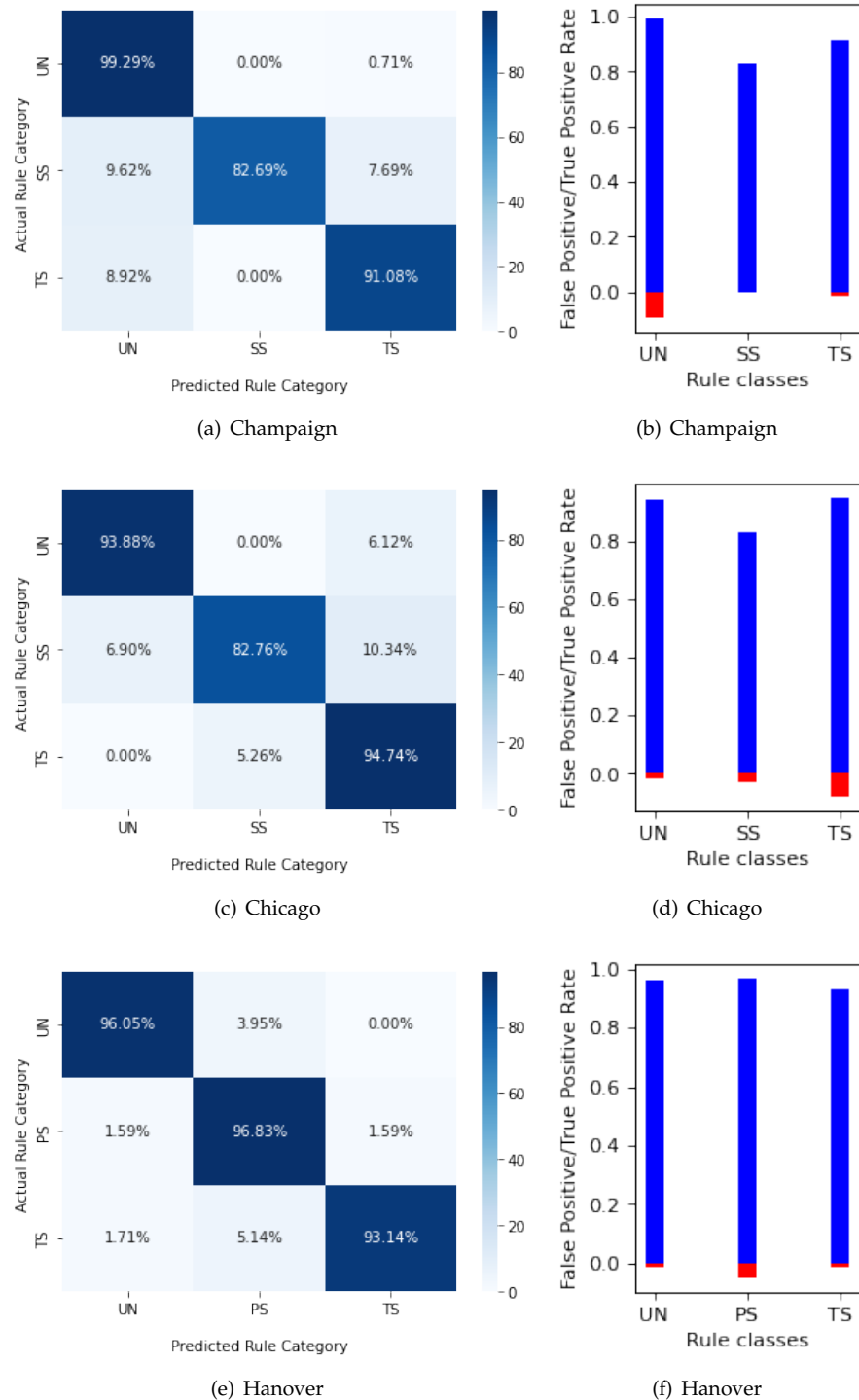
**Table 5.** Classification results of the *tuned hybrid all-static* model.

Dataset	Classifier	Label	Recall	Precision	F-Measure	Accuracy	Support
Champaign	GB	UN	0.99	0.96	0.97	0.96	424
		SS	0.83	1.0	0.90		52
		TS	0.91	0.95	0.93		157
		W.Avg.	0.96	0.96	0.96		633
Chicago	GB	UN	0.94	0.97	0.95	0.92	49
		PS	0.83	0.86	0.84		29
		TS	0.95	0.93	0.94		76
		W.Avg.	0.92	0.93	0.92		154
Hanover	GB	UN	0.96	0.91	0.93	0.96	76
		PS	0.97	0.96	0.97		315
		TS	0.93	0.97	0.95		175
		W.Avg.	0.96	0.96	0.96		566

The c-dynamic model has lower accuracy than the dynamic model. Regarding the hybrid model, we also tested two variants of the hybrid all-arm model, the *hybrid-all-static* and the *hybrid-all-dynamic*. In the first model only the dynamic features from one arm are included in the feature vector, along with the static features from all arms of the junction. In the second, only the static features from one arm are included in the feature vector, along with the dynamic features from all arms of the junction. For all datasets using the GB classifier, the hybrid-all-static model performs the same or better than the hybrid and hybrid-all-dynamic models and better than the c-dynamic, dynamic and static models. Similar results are observed for the RF classifier, except for the Hanover dataset, where the hybrid model has an accuracy of 0.92 compared to the hybrid-all-static model with an accuracy of 0.91.

Therefore, the all-arm hybrid-all-static model with the GB classifier performs better for all datasets and for this reason we select this model to use it for the experiments in the following sections. In addition, we performed feature selection and tuning of the classifier. In the Appendix A we provide plots showing the importance of the features. Interestingly, the most important features differ from dataset to dataset, even between the Champaign and Chicago datasets that share the same traffic regulator categories (UN, SS, TS). E.g., in Champaign there are more important features related to deceleration compared to Chicago, while in Chicago the important features are more related to speed percentiles along with map features. Common significant features for all datasets are the pattern features ( $p1$ ,  $p2$ ,  $p3$  and  $p4$ ). The classification results and confusion matrices for the three datasets after feature selection and tuning are presented in Table 5 and Figure 8.

As we can see in Table 5, feature selection and classifier tuning increased the accuracy by 1%, from 0.95 to 0.96 for the Champaign and Hanover datasets, and from 0.91 to 0.92 for the Chicago dataset. In Champaign and Chicago, the stop sign (SS) category is predicted slightly worse than the other two categories (F-Measure in Champaign: 0.90 (SS), 0.97 (UN), and 0.93 (TS), and in Chicago: 0.84 (SS), 0.95 (UN), and 0.94 (TS)). In Hanover, the per-class F-Measures are similar for the three classes. This observation is highlighted in the confusion matrices in Figure 8, which visually depicts the actual versus predicted classes. In the same Figure, there are also graphs of the false positive rate (FPR) and true positive rate (TPR). We can see in Figures 8(b), (d) and (f) that the highest FPRs in the three datasets are observed in different classes: UN in Champaign (0.09), TS in Chicago (0.077) and PS in Hanover (0.048). Also, the highest TPRs are observed in the same classes as the highest FPRs: Champaign: 0.99 (UN), Chicago: 0.93 (TS) and Hanover: 0.97 (PS).



**Figure 8.** Confusion matrices and false/true positive rates for the three datasets.

The lower performance in Chicago (accuracy of 0.92) compared to Champaign and Hanover (0.96 and 0.96) can be explained by the fact that the Chicago dataset is significantly smaller than the other datasets (154 regulators versus 633 (Champaign) and 566 (Hanover)), which limits the training possibilities. In addition, as already mentioned in 2.1.2, the sampling rate in Chicago is lower than the other two datasets (average 0.28Hz vs. 1Hz), which may affect the computation of the feature calculation (short-term detected events).

480  
481  
482  
483  
484  
485



### 3.2. Testing the Effect of Turning Trajectories

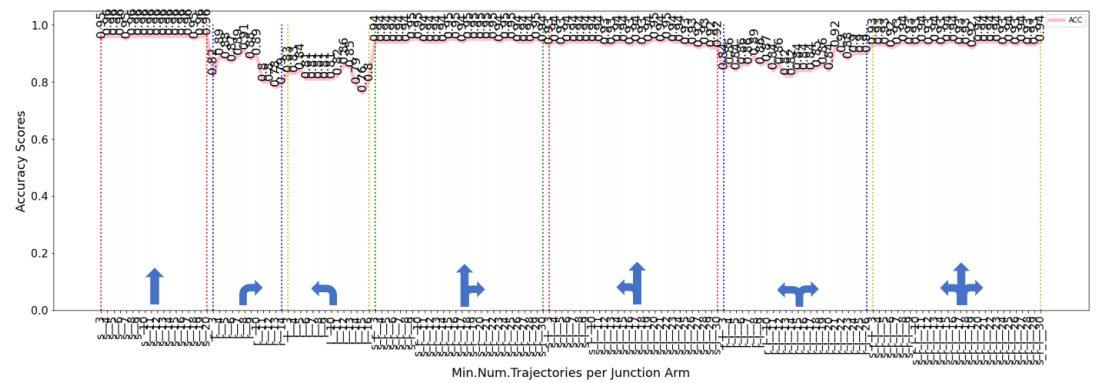
In Figure 9, we report the classification performance for the three datasets under different traversal settings: crossing direction and number of trajectories per intersection arm. In the first case, we examine whether considering samples moving only straight ahead positively affects classification, assuming that turning behavior affects speed, so excluding curved trajectories can eliminate their bias. In the second case, we seek whether there is an optimal number of trajectories that an intersection arm should have during training and thus exclude from the training dataset intersections with fewer trajectories than this number.

We looked at all possible turning settings and their combinations. Regarding the labels on the horizontal axis of Figure 9,  $s\_$  refers only to straight trajectories,  $r\_$  to trajectories that turn right,  $l\_$  to trajectories that turn left,  $s\_r\_$  refers to straight/right turn,  $s\_l\_$  to straight/left turn,  $r\_l\_$  to right/left turn and  $s\_r\_l\_$  to straight/right/left trajectories. The number after these prefixes refers to the number  $n$  of trajectories used to select the intersection arms (minimum number of trajectories per intersection arm) and to calculate the motion patterns (i.e. motion patterns observed on an intersection arm are calculated by summarizing the behaviour of at least  $n$  trajectories that cross it). Not all turning settings are tested with the same number of trajectories, because for each turning/crossing setting, we require the test data set to contain at least 7 junction arms per class. E.g., in the Champaign dataset (Figure 9(a)), we tested the straight trajectories for various numbers 3, 4, ..., 20, because for minimum number of trajectories equal to 21, the number of junction arms in the test set (10-fold cross-validation) could not contain more than 7 stop controlled (SS) junction arms.

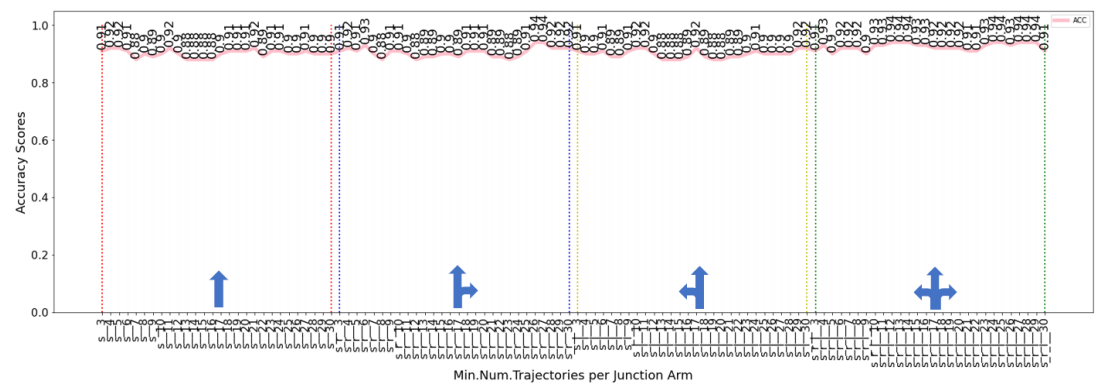
Regarding the effect of turning trajectories on classification performance, we see that using right, left or right/left traces has lower performance than using straight traces and combinations of straight and turning traces (Figure 9(a) and (c)). When using a combination of straight and turning traces, we cannot see a strong negative effect, but this can perhaps be explained by the fact that there are significantly more straight crossings than left and right in the dataset (in Champaign 20514 straight, 2619 right and 2768 left, in Hanover 19092 straight, 3394 right and 3073 left, in Chicago 12638 straight, 2820 right and 1301 left).

A possible explanation for the poor classification performance when using exclusively turning trajectories (e.g. in Champaign 0.88 accuracy for right turning trajectories when junction arms have at least 5 trajectories -see  $r\_5$  at Figure 9(a)), is the smaller dataset used for training compared to the other settings. Table 6 shows the number of train/test junction arms per control type in Champaign dataset, when each arm has at least five trajectories. We see that the training sets when only right, left or right/left trajectories are used are much smaller than the set with straight trajectories. Additionally the fact that features are computed from a few trajectories (as the number of right and left trajectories per junction arm is small), may also explain the bad performance. Therefore, one reason for the poor performance may be related to the dataset itself (which affects the feature computation and the size of the training dataset) and not solely to the condition we consider here (drive straight through an intersection or turn). Perhaps the same analysis on a dataset with numerically more junction arms sampled from a higher density of turning trajectories would not show a negative effect of turning trajectories.

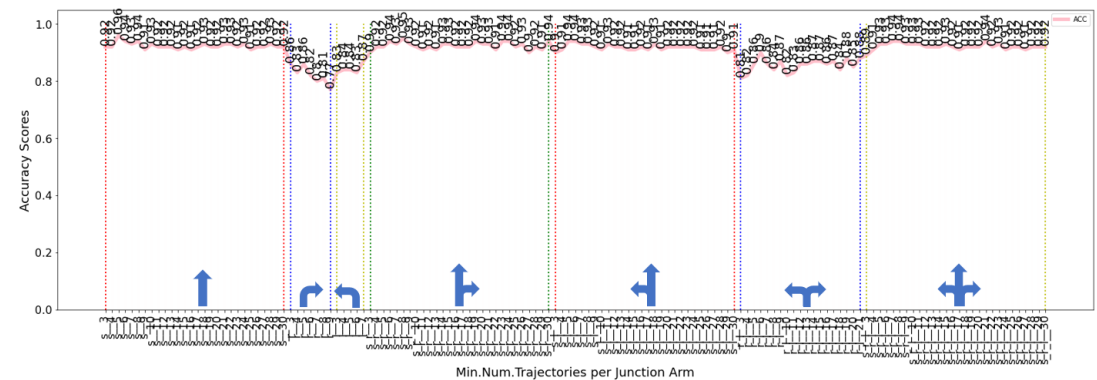
Another observation is that on the Champaign dataset the performance when only straight trajectories are used is better than when straight and turning trajectories are used. E.g., with at least 5 straight trajectories the accuracy is 96% and with at least 5 straight/turning trajectories the accuracy is 93% (Figure 9(a)). On the Hanover dataset (Figure 9(c)), on average the performance when using only straight trajectories is better than when using straight and turning trajectories, but the effect is less strong than in the Champaign dataset. For both datasets, the difference in accuracy between using straight and all trajectories (straight and curved) is between 1%-3%. Therefore, in both datasets, excluding curved trajectories has a positive effect on classification performance and the optimal number of straight trajectories is 5.



(a) Champaign.



(b) Chicago.



(c) Hanover.

**Figure 9.** Experiments with different turning settings ( $s_:$  straight trajectories,  $r_:$  right turning trajectories,  $l_:$  left turning,  $s_r_:$  straight and right turning,  $s_l_:$  straight and left turning,  $r_l_:$  right and left turning,  $s_r_l_:$  straight, right and left turning trajectories).

However, in the Chicago dataset the same observation does not hold, i.e. with at least 15 straight trajectories the accuracy is 88% and with at least 15 straight/turning trajectories the accuracy is 94% (Figure 9(b)). Although there are not a sufficient number of crossing arms crossed by turning trajectories for training and testing as in the other two datasets, a possible explanation for the slightly increased performance when all trajectories are used is that the training dataset becomes larger when straight/turning trajectories are used than when only straight trajectories are used, so the classifier learns better. E.g. when at least 5 straight trajectories are used, the dataset contains 49 UN, 29 SS and 76 TS, while when at

540  
541  
542  
543  
544  
545  
546  
547

least 5 tracks (turning and straight tracks) are used the dataset contains 50 UN, 40 SS and 115 TS.

Therefore, judging from the two larger datasets that have consistent results, we can conclude that curved tracks at intersections affect the classification by about 1%-3% in accuracy and therefore for the next experiments we will only use straight tracks (minimum number of tracks per junction arm equal to 5), excluding all curved tracks at intersections, similar to what [Hu et al. \[18\]](#) and [Golze et al. \[21\]](#) do in their studies.

**Table 6.** Dataset size for different trajectory direction settings with minimum number of trajectories per junction arm equal to 5 (Champaign dataset).

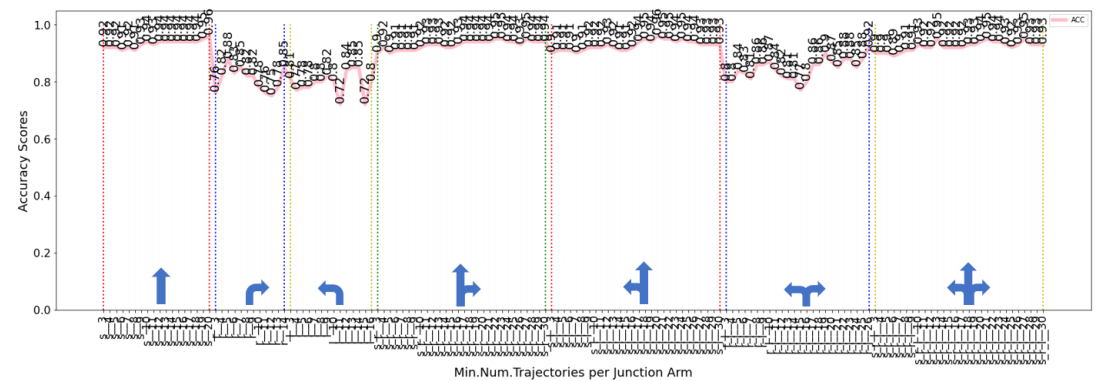
Trajectory Direction	Rule	Junction arms	Classification Accuracy
Straight	UN	424	0.96
	SS	52	
	TS	157	
	Sum	633	
Right	UN	26	0.88
	SS	29	
	TS	59	
	Sum	114	
Left	UN	36	0.81
	SS	18	
	TS	48	
	Sum	102	
Right/Left	UN	71	0.84
	SS	59	
	TS	108	
	Sum	238	
All	UN	457	0.93
	SS	100	
	TS	192	
	Sum	749	

### 3.3. Testing the Effect of Number of Trajectories on Classification Performance

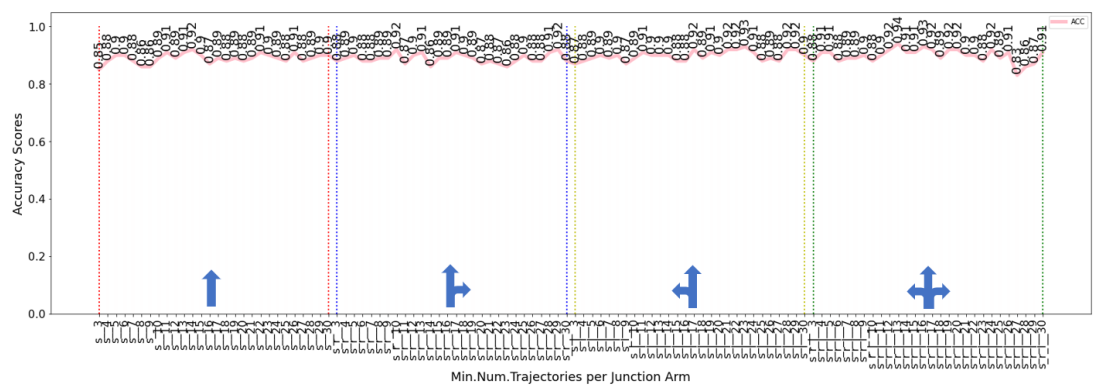
The number of trajectories used as a minimum requirement for the calculation of classification features, as well as for the selection of junction arms for training and testing, seems to affect performance. Comparing the case of using a *certain* number of trajectories, i.e., a subset of all trajectories crossing a junction arm, with the case of using *all* available trajectories per junction arm (as explained in Section 2.2.4.2), the latter case achieves better performance on average. This result holds for all datasets (compare Figure 9 with Figure 10) and seems reasonable, as the more trajectories are used to compute the classification features (which are statistical values of physical features, as explained in Section 2.2.2), the better the latter reflect the actual movement behavior.

On the Champaign dataset we see from Figure 10(a) that with just 3 straight trajectories per junction arm, we can achieve 92% accuracy. Increasing the number of trajectories also increases the classification performance. The best result, 96%, is achieved with 20 trajectories. However, when we compare the results with those in Figure 9(a), we see that limiting the number of trajectories per arm yields lower classification performance. E.g. with *at least* 4 straight trajectories per junction arm the accuracy is 96%.

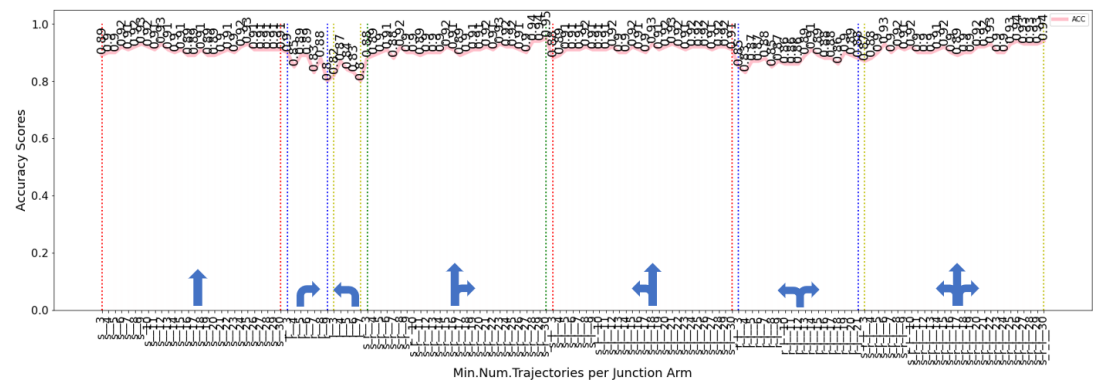
In the Hanover dataset, we can also see from Figure 9(c) that when all available traces are used the performance is better than using a certain number (Figure 10(c)). For accuracy above 91% at least 7 traces per junction arm are required (Figure 10(c)). The best accuracy 93% is achieved with 9 trajectories. In contrast, by allowing the use of all available



(a) Champaign.



(b) Chicago.



(c) Hanover.

**Figure 10.** Experiments with different number of trajectories where classification features are computed using a *certain* number of trajectories, i.e. 3, 4, ..., and not all available crossing trajectories.

trajectories per arm, with at least 3 traces per arm, the accuracy is always equal to or greater than 91%, and the best accuracy of 96% is achieved with at least 5 trajectories (Figure 9(c)).

On the Chicago dataset, the same result is also observed with the other two datasets. Using a *certain* number of trajectories per arm gives a lower classification accuracy than when all available trajectories are used (Figure 10(b) vs. Figure 9(b)). However, with only 3 straight trajectories per arm the accuracy is always equal to or greater than 85% and with only 4 straight trajectories equal to or greater than 86% (Figure 10(b)).

Therefore, for feature computation, excluding the number of trajectories to a certain number negatively affects the classification performance. However, with only 3 straight trajectories per junction arm, the classification accuracy is equal to or greater than 85%

575  
576  
577  
578  
579  
580  
581  
582  
583  
584

**Table 7.** Classification results of the *tuned hybrid all-static* model after applying a consistency check of domain knowledge rules.

Dataset	Classifier	Label	Recall	Precision	F-Measure	Accuracy	Support
Champaign	GB	UN	0.96	0.99	0.97	0.97	427
		SS	0.97	0.98	0.97		284
		TS	0.98	0.91	0.94		237
		W.Avg.	0.97	0.97	0.96		948
Chicago	GB	UN	0.96	0.94	0.95	0.94	49
		PS	0.92	0.86	0.89		42
		TS	0.94	0.97	0.95		109
		W.Avg.	0.94	0.94	0.94		200
Hanover	GB	UN	0.94	0.98	0.96	0.97	123
		PS	0.97	0.97	0.97		315
		TS	0.98	0.96	0.97		281
		W.Avg.	0.97	0.97	0.97		719

across all datasets (85% in Chicago, 89% in Hanover and 92% in Champaign). Also, with only 5 straight trajectories the accuracy is equal to or greater than 90% (90% in Chicago and 92% in Champaign and Hanover).

### 3.4. Application of Domain Knowledge Rules

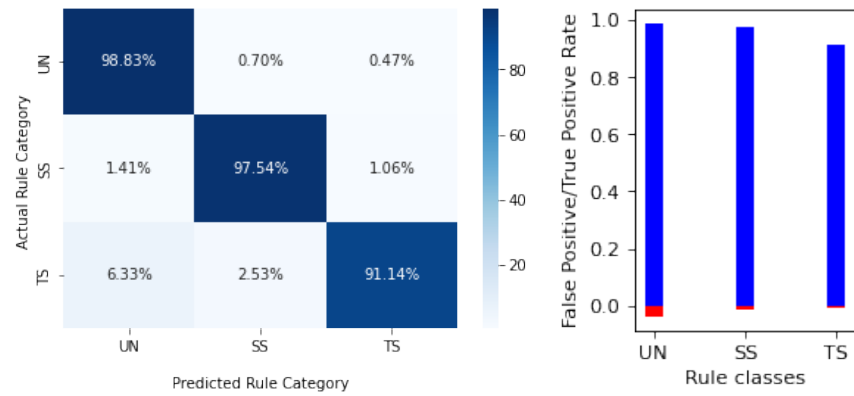
The Table 7 shows the classification report after applying the domain knowledge rules consistency check, as explained in Section 2.3. Figure 11 also shows the confusion matrices along with the FPR/TPR graphs for the three datasets.

The first observation from Table 7 is that accuracy increases by 1% in Champaign and Hanover (from 96% to 97%) and by 2% in Chicago (from 92% to 94%). Also, by comparing the number of predictions (compare Support column in Table 5 and Table 7), we can see that in Champaign we have 315 predictions from arms with missing data, which means that not only were predictions made that are equal to 50% of the original dataset with no data (original dataset: 633 arms), but these predictions are correct (accuracy increased to 1%). In Chicago, 46 predictions were made on arms with missing data, corresponding to 29.9 % of the original dataset. Similarly, in Hanover, 153 regulators were predicted from arms with missing data, corresponding to 27% of the original dataset.

In terms of FPRs, in Champaign the FPRs for UN, SS and TS are: 4%, 1.4%, 0.7% respectively and in Chicago 1.3%, 1.9% and 7.7%. In Hanover the FPRs for UN, SS and TS are 1.3%, 2.2%, 1.1%. Compared to the FPRs of Figure 8, in Champaign the FPR for UN decreased from 9% to 4% (55.6% decrease), in Chicago the FPR for TS remained the same, but for SS decreased from 4% to 2% (50% decrease), and in Hanover the FPR for PS decreased from 4.8% to 2.2% (54% decrease). Moreover, the average FPR (TPR) across regulator classes decreases (increases) in Champaign from 3.5% (91%) to 1.9% (96%), in Chicago from 4.8% (89%) to 3.6% (92%) and in Hanover from 2.6% (95%) to 1.6% (97%).

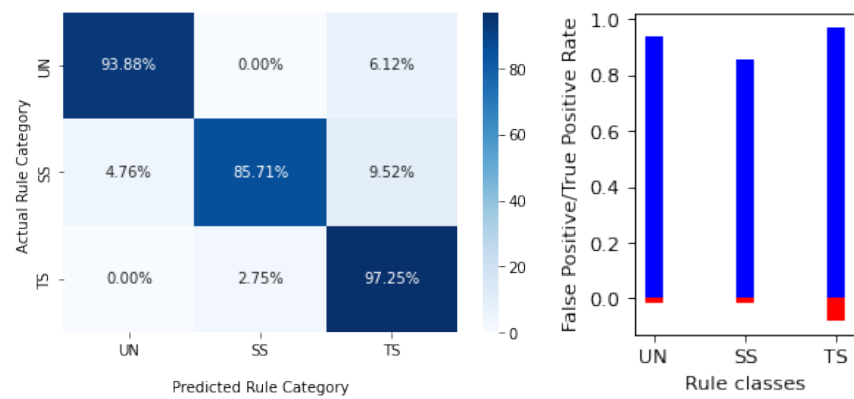
Therefore, by applying domain knowledge rules, there is a gain in accuracy between 1% and 2%, but more importantly, accurate predictions can be made for arms with incomplete data corresponding to 27%-50% of the original data. Furthermore, the FPR of the class with the highest FPR either remains the same (Chicago) or decreases to 50% (Champaign and Hanover), validating our proposal to use domain knowledge rules for both recovering misclassified arms and predicting regulators for arms without data.





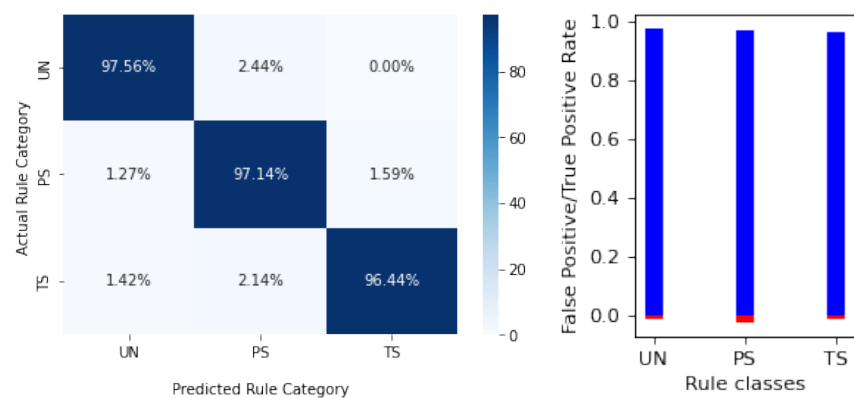
(a) Champaign

(b) Champaign



(c) Chicago

(d) Chicago



(e) Hanover

(f) Hanover

**Figure 11.** Confusion matrices and false/true positive rates for the three datasets after applying consistency check using domain knowledge rules.

#### 4. Discussion

The main findings of this research paper are the following:

1. The traffic rule recognition method proposed in this paper, which combines data from trajectories and OSM, can provide accurate results for rule sets consisting of controlled intersections of UN, SS, PS, and UN: 97% accuracy in Champaign and Hanover, and 94% in the smaller Chicago dataset.
2. Including information in the feature vector from adjacent junction arms proved beneficial for classification: all-arm models outperformed single-arm models.
3. The negative effect on accuracy, as validated by the two larger datasets, when both straight and curved trajectories are used, was found to be between 1%-3%. Therefore, the exclusion of curved trajectories has a positive effect on classification performance.
4. The optimal number of straight tracks is five.
5. Excluding the number of tracks to a certain number negatively affects the classification performance. However, with only three straight trajectories per junction arm, classification accuracy is equal to or greater than 85% across all datasets (85% in Chicago, 89% in Hanover and 92% in Champaign). With only five straight trajectories, accuracy is equal to or greater than 90% (90% in Chicago and 92% in Champaign and Hanover).
6. By applying domain knowledge rules, there is a gain in accuracy between 1% and 2%, but more importantly, accurate predictions can be made for arms with no data, corresponding to 27%-50% of the original data. Moreover, the average FPR (TPR) across all classes of regulators decreases (increases) between 1.9%-4.8% (2%-5%), so our proposal to use domain knowledge rules for both recovering misclassified arms and for predicting regulators from arms without data, is validated.

We find it more interesting to extend this study to how classification can predict rules with high accuracy on limited labeled data. As discussed in 2.1, labeling intersections is a time-consuming task, and although there are many trajectory datasets that one can use for TRR, the need for labeled data for training and testing makes the latter unsuitable for this purpose. One idea is to explore unsupervised methods where no labelled data is required. A second idea is to explore semi-supervised methods, such as label propagation and self-learning, where only limited labeled data is used to train a classifier. A third idea is to explore the possibility of transferring learning from one city to another, i.e., training a classifier with labeled data from a city  $X$  and testing in a city  $Y$ , assuming no labeled data from the latter city.

In addition, an important aspect of the problem that needs to be considered is whether the proposed approach would perform equally well in smaller cities, where driving behaviour is influenced by factors other than traffic regulations, e.g. pedestrians who, knowing that vehicles are moving at low speed, cross intersections more freely.

Moreover, it would be interesting to test the performance of the proposed hybrid model with missing OSM data. In the three datasets we tested, very often speed-related data were missing from OSM, but the performance remained high. It would be interesting to investigate under what conditions of missing data, performance would be affected to the extent that, for example, the dynamic model would be preferable to the hybrid model.

Furthermore, another important aspect is to investigate to what extent the sampling rate of GPS traces affects classification performance. We saw that the Chicago dataset has one third of the sampling rate of the other two datasets and the accuracy is 3% worse than them. The question is whether this difference is due to the lower sampling rate or the size of the dataset (Chicago is much smaller than the other datasets). One idea to test this is to undersample the Champaign and Hanover datasets so that the vehicle location is sampled at about 0.3 Hz (as in Chicago) and compare the classification accuracy.

Finally, the way in which variability of trajectory densities affect the classification would be another parameter of the TRR problem that deserves further investigation. Since not all intersections attract the same traffic, the datasets are irregular, e.g. a section of a city is sampled from dozens of trajectories and other parts from only a few tracks. This aspect is not taken into account in the current settings of splitting the datasets into training and

test sets. Perhaps splitting the dataset taking this aspect into account would provide even better results.

## 5. Conclusions

In this paper, a new method for identifying traffic regulations from GPS trajectories is proposed. A modification of a well-known clustering algorithm for detecting stopping and deceleration events was presented. By detecting such driving events, we categorize intersections into four traffic classes, which, together with other statistical values of the detected events and the average vehicle traverse speed at these locations, describe the driving behavior at the regulated locations (*dynamic* classification model). Mixing in the dynamic model static features extracted from the OSM (*static* model), leads to a *hybrid* model that was shown to have better classification performance than the other two models. For each of the three models, two variants of the feature vector were tested, one where only features associated with a single junction arm are used (*one-arm* model) and another where features also from neighbouring junction arms of the same junction are used to classify one arm (*all-arm* model). The hybrid all-arm model provided the best classification accuracy on the three datasets used to test the methodology, 94% on the smallest dataset and 97% on the other two datasets. The minimum optimal number of trajectories crossing the intersections was found to be five (straight trajectories). The exclusion of curved trajectories from the feature calculation was found to have a positive effect on classification performance. Finally, by applying a set of domain knowledge rules to the predicted labels, we were able to both recover misclassified intersection arms and predict labels from arms with no data, corresponding to 27%-50% of the original dataset, while further increasing classification accuracy by 1%-2%. New research directions were proposed based on the limitations of the study, discussing also ideas that can better clarify these issues.

## Disclaimer

This paper was prepared for information purposes with contributions from the Future Lab for Applied Research and Engineering (FLARE) group of JPMorgan Chase Bank, N.A.. This paper is not a product of the Research Department of JPMorgan Chase Bank, N.A. or its affiliates. Neither JPMorgan Chase Bank, N.A. nor any of its affiliates make any explicit or implied representation or warranty and none of them accept any liability in connection with this paper, including, but limited to, the completeness, accuracy, reliability of information contained herein and the potential legal, compliance, tax or accounting effects thereof. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction.

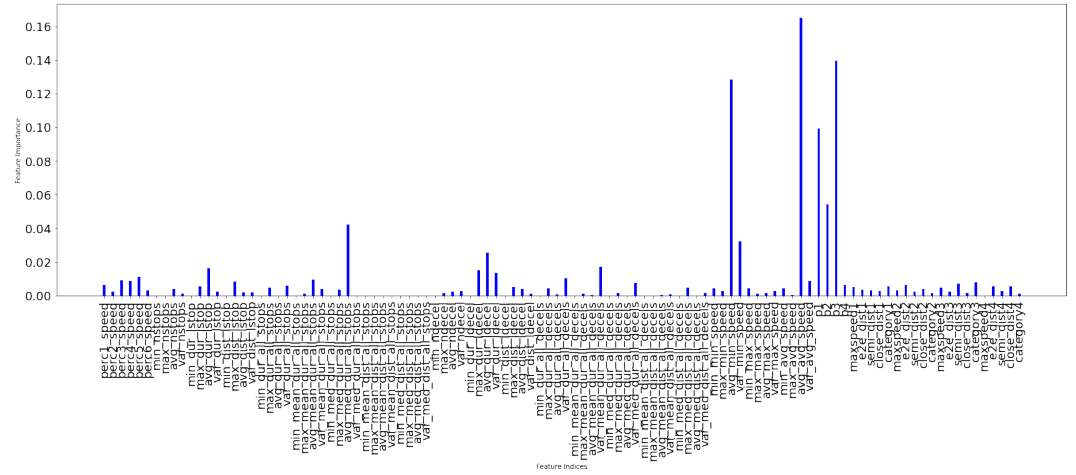
## Appendix A

The appendix is an optional section that can contain details and data supplemental to the main

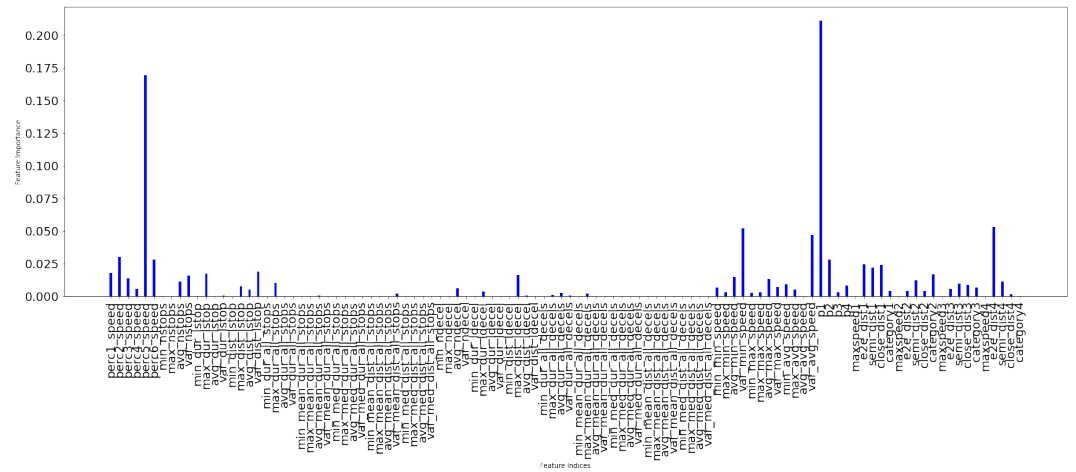
705

706

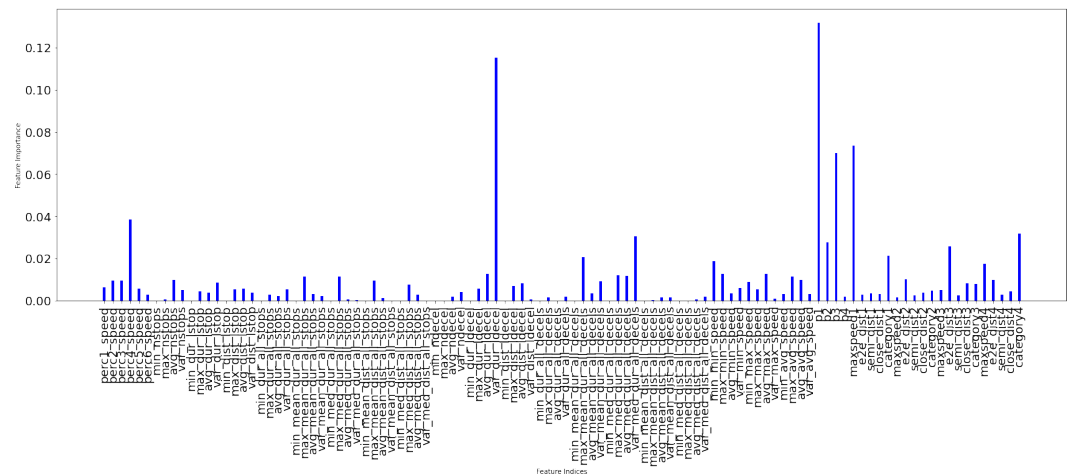
707



(a) Champaign



(b) Chicago



(c) Hanover

Figure A1. Feature importance for the the three dataset.

**Author Contributions:** Conceptualization, S.Z., M.S. and S.H.; methodology, S.Z.; software, S.Z.; validation, S.Z. and M.S.; formal analysis, S.Z.; investigation, S.Z.; resources, S.Z. and S.H.; data curation, S.Z. and S.H.; writing—original draft preparation, S.Z.; writing—review and editing, S.Z. and M.S.; visualization, S.Z.; supervision, M.S.; project administration, S.Z. and M.S.; funding acquisition, M.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the German Research Foundation (Deutsche Forschungsgemeinschaft (DFG)) with grant number 227198829/GRK1931.

**Informed Consent Statement:** Informed consent was obtained from all subjects involved in the study.

**Data Availability Statement:** The groundtruth map from the Chicago dataset, is accessible in a public repository [27] and has an open data licence. The Chicago trajectory dataset, as already mentioned, is publicly available in [39]. The Hannover and Champaign datasets are not available due to privacy restrictions as the gps trajectories could compromise the privacy of the research participants.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## Abbreviations

The following abbreviations are used in this manuscript:

OSM	OpenStreetMap
GPS	Global Positioning System
SC	Spatial Crowdsourcing
CC	Crowdsourcing
DLSTM	Distributed Long Short Term Memory
ROC	Receiver Operating Characteristic
AUC	Area Under the ROC Curve
CVAE	Conditional Variational Autoencoder
TRR	Traffic Regulator Recognition
TS	Traffic Signals
SS	Stop Sign
PS	Priority Sign
YS	Yield Sign
RF	Random Forest
GB	Gradient Boosting
FPR	False Positive Rate
TPR	True Positive Rate

## References

- Goodchild, M. Citizens as sensors: the world of volunteered geography. *GeoJournal* **2007**, *69*, 211–221. <https://doi.org/10.1007/s10708-007-9111-y>.
- Gummidi, S.R.B.; Xie, X.; Pedersen, T.B. A Survey of Spatial Crowdsourcing. *ACM Trans. Database Syst.* **2019**, *44*. <https://doi.org/10.1145/3291933>.
- Heipke, C. Crowdsourcing geospatial data. *ISPRS Journal of Photogrammetry and Remote Sensing* **2010**, *65*, 550 – 557. ISPRS Centenary Celebration Issue, <https://doi.org/10.1016/j.isprsjprs.2010.06.005>.
- Tang, J.; Deng, M.; Huang, J.; Liu, H.; Chen, X. An Automatic Method for Detection and Update of Additive Changes in Road Network with GPS Trajectory Data. *ISPRS International Journal of Geo-Information* **2019**, *8*. <https://doi.org/10.3390/ijgi8090411>.
- Shan, Z.; Wu, H.; Sun, W.; Zheng, B. COBWEB: A Robust Map Update System Using GPS Trajectories. In Proceedings of the Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing; Association for Computing Machinery: New York, NY, USA, 2015; UbiComp '15, p. 927–937. <https://doi.org/10.1145/2750858.2804286>.
- Fox, A.; Kumar, B.V.; Chen, J.; Bai, F. Multi-lane pothole detection from crowdsourced undersampled vehicle sensor data. *IEEE Transactions on Mobile Computing* **2017**, *16*, 3417–3430.
- Wage, O.; Sester, M. JOINT ESTIMATION OF ROAD ROUGHNESS FROM CROWD-SOURCED BICYCLE ACCELERATION MEASUREMENTS. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **2021**, *V-4-2021*, 89–96. <https://doi.org/10.5194/isprs-annals-V-4-2021-89-2021>.



8. Vij, D.; Aggarwal, N. Smartphone based traffic state detection using acoustic analysis and crowdsourcing. *Applied Acoustics* **2018**, *138*, 80–91. <https://doi.org/10.1016/j.apacoust.2018.03.029>. 745 746
9. Minson, S.E.; Brooks, B.A.; Glennie, C.L.; Murray, J.R.; Langbein, J.O.; Owen, S.E.; Heaton, T.H.; Iannucci, R.A.; Hauser, D.L. Crowdsourced earthquake early warning. *Science Advances* **2015**, *1*. <https://doi.org/10.1126/sciadv.1500036>. 747 748
10. Salpietro, R.; Bedogni, L.; Di Felice, M.; Bononi, L. Park Here! a smart parking system based on smartphones' embedded sensors and short range Communication Technologies. In Proceedings of the 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 2015, pp. 18–23. 749 750 751
11. Zhou, X.; Zhang, L. Crowdsourcing functions of the living city from Twitter and Foursquare data. *Cartography and Geographic Information Science* **2016**, *43*, 393–404. <https://doi.org/10.1080/15230406.2015.1128852>. 752 753
12. Alshayeb, S.; Stevanovic, A.; Effinger, J.R. Investigating impacts of various operational conditions on fuel consumption and stop penalty at signalized intersections. *International Journal of Transportation Science and Technology* **2021**. <https://doi.org/10.1016/j.ijst.2021.09.005>. 754 755 756
13. Gastaldi, M.; Meneguzzo, C.; Rossi, R.; Lucia, L.D.; Gecchele, G. Evaluation of Air Pollution Impacts of a Signal Control to Roundabout Conversion Using Microsimulation. *Transportation Research Procedia* **2014**, *3*, 1031–1040. 17th Meeting of the EURO Working Group on Transportation, EWGT2014, 2-4 July 2014, Sevilla, Spain, <https://doi.org/10.1016/j.trpro.2014.10.083>. 757 758 759
14. OpenStreetMap contributors. Planet dump retrieved from <https://planet.osm.org>. <https://www.openstreetmap.org>, 2020. Accessed: 2020-08-17. 760 761
15. Zourlidou, S.; Sester, M. Traffic Regulator Detection and Identification from Crowdsourced Data—A Systematic Literature Review. *ISPRS International Journal of Geo-Information* **2019**, *8*. <https://doi.org/10.3390/ijgi8110491>. 762 763
16. Huang, S.; Lin, H.; Chang, C. An in-car camera system for traffic sign detection and recognition. In Proceedings of the 2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS), 2017, pp. 1–6. 764 765 766
17. Ardianto, S.; Chen, C.; Hang, H. Real-time traffic sign recognition using color segmentation and SVM. In Proceedings of the 2017 International Conference on Systems, Signals and Image Processing (IWSSIP), 2017, pp. 1–5. 767 768
18. Hu, S.; Su, L.; Liu, H.; Wang, H.; Abdelzaher, T.F. SmartRoad: Smartphone-Based Crowd Sensing for Traffic Regulator Detection and Identification. *ACM Trans. Sen. Netw.* **2015**, *11*, 55:1–55:27. <https://doi.org/10.1145/2770876>. 769 770
19. Merry, K.; Bettinger, P. Smartphone GPS accuracy study in an urban environment. *PLoS One* **2019**, *14*. <https://doi.org/10.1371/journal.pone.0219890>. 771 772
20. Saremi, F.; Abdelzaher, T.F. Combining Map-Based Inference and Crowd-Sensing for Detecting Traffic Regulators. *2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems* **2015**, pp. 145–153. 773 774
21. Golze, J.; Zourlidou, S.; Sester, M. Traffic Regulator Detection Using GPS Trajectories. *KN - Journal of Cartography and Geographic Information* **2020**. <https://doi.org/10.1007/s42489-020-00048-x>. 775 776
22. Liao, Z.; Xiao, H.; Liu, S.; Liu, Y.; Yi, A. Impact Assessing of Traffic Lights via GPS Vehicle Trajectories. *ISPRS International Journal of Geo-Information* **2021**, *10*. <https://doi.org/10.3390/ijgi10110769>. 777 778
23. Méneroux, Y.; Guilcher, A.; Saint Pierre, G.; Hamed, M.; Mustiere, S.; Orfila, O. Traffic signal detection from in-vehicle GPS speed profiles using functional data analysis and machine learning. *International Journal of Data Science and Analytics* **2020**, *10*, 101–119. <https://doi.org/10.1007/s41060-019-00197-x>. 779 780 781
24. Cheng, H.; Zourlidou, S.; Sester, M. Traffic Control Recognition with Speed-Profiles: A Deep Learning Approach. *ISPRS International Journal of Geo-Information* **2020**, *9*. <https://doi.org/10.3390/ijgi9110652>. 782 783
25. Ahmed, M.; Karagiorgou, S.; Pfoser, D.; Wenk, C. A comparison and evaluation of map construction algorithms using vehicle tracking data. *GeoInformatica* **2015**, *19*, 601–632. <https://doi.org/10.1007/s10707-014-0222-6>. 784 785
26. Mapillary. Mapillary: a street-level imagery platform. <https://www.mapillary.com/>, 2022. Accessed: 2022-04-20. 786
27. Zourlidou, S.; Golze, J. Ground-truth Intersection Regulators for Chicago **2022**. <https://doi.org/10.25835/0vifyzqi>. 787
28. Palma, A.T.; Bogorny, V.; Kuijpers, B.; Alvares, L.O. A Clustering-based Approach for Discovering Interesting Places in Trajectories. In Proceedings of the Proceedings of the 2008 ACM Symposium on Applied Computing; ACM: New York, NY, USA, 2008; SAC '08, pp. 863–868. <https://doi.org/10.1145/1363686.1363886>. 788 789 790
29. Comito, C.; Falcone, D.; Talia, D. Mining human mobility patterns from social geo-tagged data. *Pervasive and Mobile Computing* **2016**, *33*, 91–107. <https://doi.org/10.1016/j.pmcj.2016.06.005>. 791 792
30. Niu, X.; Wang, S.; Wu, C.Q.; Li, Y.; Wu, P.; Zhu, J. On a clustering-based mining approach with labeled semantics for significant place discovery. *Information Sciences* **2021**, *578*, 37–63. <https://doi.org/https://doi.org/10.1016/j.ins.2021.07.050>. 793 794
31. Spaccapietra, S.; Parent, C.; Damiani, M.L.; de Macedo, J.A.; Porto, F.; Vangenot, C. A Conceptual View on Trajectories. *Data Knowl. Eng.* **2008**, *65*, 126–146. <https://doi.org/10.1016/j.datak.2007.10.008>. 795 796
32. Kang, J.H.; Welbourne, W.; Stewart, B.; Borriello, G. Extracting Places from Traces of Locations. In Proceedings of the Proceedings of the 2Nd ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots; ACM: New York, NY, USA, 2004; WMASH '04, pp. 110–118. <https://doi.org/10.1145/1024733.1024748>. 797 798 799
33. Feuerhake, U.; Kuntzsch, C.; Sester, M. Finding interesting places and characteristic patterns in spatio-temporal trajectories. In Proceedings of the 8th LBS symposium, 2011. 800 801
34. Wu, T.; Shen, H.; Qin, J.; Xiang, L. Extracting Stops from Spatio-Temporal Trajectories within Dynamic Contextual Features. *Sustainability* **2021**, *13*. <https://doi.org/10.3390/su13020690>. 802 803



- 
35. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Proceedings of the Proc. 2<sup>nd</sup> Int. Conf. on Knowledge Discovery and Data Mining(KDD '96), 1996, pp. 226–231. 804
36. Tang, L.; Kan, Z.; Zhang, X.; Yang, X.; Huang, F.; Li, Q. Travel time estimation at intersections based on low-frequency spatial-temporal GPS trajectory big data. *Cartography and Geographic Information Science* **2016**, *43*, 417–426. <https://doi.org/10.1080/15230406.2015.1130649>. 805
37. Carisi, R.; Giordano, E.; Pau, G.; Gerla, M. Enhancing in vehicle digital maps via GPS crowdsourcing. In Proceedings of the 2011 Eighth International Conference on Wireless On-Demand Network Systems and Services, 2011, pp. 27–34. <https://doi.org/10.1109/WONS.2011.5720196>. 806
38. XGBoost Python. XGBoost Python Library. <https://xgboost.readthedocs.io/en/stable/python/index.html>, 2022. Accessed: 2022-02-15. 807
39. Ahmed, M.; Karagiorgou, S.; Pfoser, D.; Wenk, C. A comparison and evaluation of map construction algorithms using vehicle tracking data. *GeoInformatica* **2014**, *19*, 601–632. <https://doi.org/10.1007/s10707-014-0222-6>. 808
- 809
- 810
- 811
- 812
- 813
- 814
- 815