

SmartRoad: Smartphone-Based Crowd Sensing for Traffic Regulator Detection and Identification

SHAOHAN HU, University of Illinois at Urbana-Champaign

LU SU, State University of New York at Buffalo

HENGCHANG LIU, University of Science and Technology of China

HONGYAN WANG and TAREK F. ABDELZAHER, University of Illinois at Urbana-Champaign

In this article we present SmartRoad, a crowd-sourced road sensing system that detects and identifies traffic regulators, traffic lights, and stop signs, in particular. As an alternative to expensive road surveys, SmartRoad works on participatory sensing data collected from GPS sensors from in-vehicle smartphones. The resulting traffic regulator information can be used for many assisted-driving or navigation systems. In order to achieve accurate detection and identification under realistic and practical settings, SmartRoad automatically adapts to different application requirements by (i) intelligently choosing the most appropriate information representation and transmission schemes, and (ii) dynamically evolving its core detection and identification engines to effectively take advantage of any external ground truth information or manual label opportunity. We implemented SmartRoad on a vehicular smartphone test bed, and deployed it on 35 external volunteer users' vehicles for two months. Experiment results show that SmartRoad can robustly, effectively, and efficiently carry out the detection and identification tasks.

Categories and Subject Descriptors: C.5.3 [Computer System Implementation]: Microcomputers—*Portable devices*

General Terms: Algorithms, Design, Experimentation, Performance

Additional Key Words and Phrases: Crowd sensing, road sensing, traffic regulator, smartphone

ACM Reference Format:

Shaohan Hu, Lu Su, Hengchang Liu, Hongyan Wang, and Tarek F. Abdelzaher. 2015. SmartRoad: Smartphone-based crowd sensing for traffic regulator detection and identification. *ACM Trans. Sensor Netw.* 11, 4, Article 55 (July 2015), 27 pages.

DOI: <http://dx.doi.org/10.1145/2770876>

1. INTRODUCTION

Traffic lights and stop signs play important roles in our daily lives. They regulate traffic flow by providing guidance and warnings to drivers, directly affecting their safety on the road. Serious accidents happen when drivers fail to notice signs, either due to visibility issues or their psychological states [Maldonado-Bascon et al. 2007].

This work was sponsored in part by NSF grants CNS 13-45266 and CNS 10-40380, DTRA grant HDTRA1-10-1-0120, NSFC grants GG2260080042 and BJ2260080039, and the USTC Innovation Funding DG2260010011.

Authors' addresses: S. Hu, H. Wang, and T. F. Abdelzaher, Siebel Center for Computer Science, 201 N Goodwin Avenue, Urbana IL 61801; emails: {shu17, wang44, zaher}@illinois.edu; L. Su, Department of Computer Science and Engineering, State University of New York at Buffalo, 321 Davis Hall, Buffalo, New York 14260; email: lusu@buffalo.edu; H. Liu, School of Computer Science and Technology, University of Science and Technology of China, 188 Ren Ai Road, Office 406-4 Suzhou, Jiangsu 215123, China; email: hliu@ustc.edu.cn.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2015 ACM 1550-4859/2015/07-ART55 \$15.00

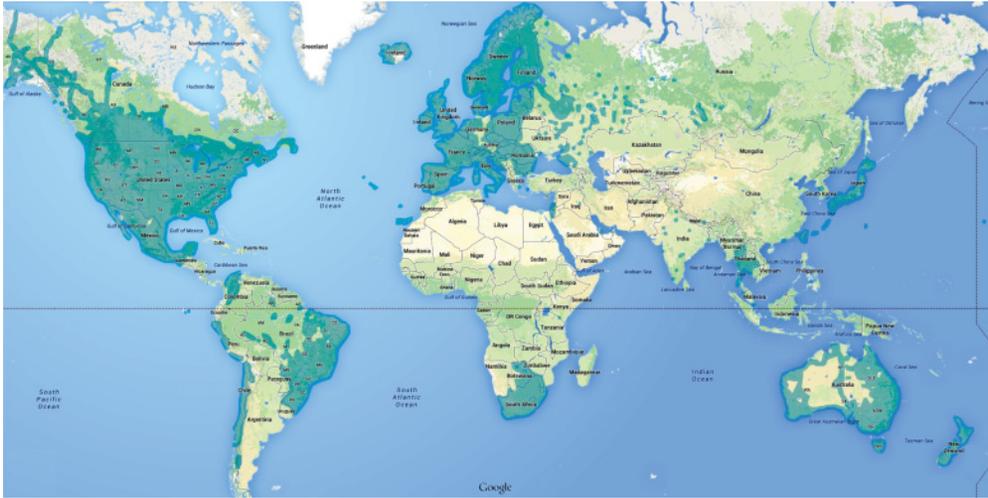
DOI: <http://dx.doi.org/10.1145/2770876>

Driver-Assistance Systems (or DAS) can help improve road safety, for example, by properly warning a distracted driver of the upcoming stop sign. Therefore, having detailed location information of traffic regulators can help improve drivers' safety on the road. On the other hand, despite their safety benefits, traffic regulators do take their toll—the stop-and-go movement pattern of vehicles have resulted in a substantial increase of gas consumption and CO₂ emissions [Audi Travolution 2010]. Recent efforts have been taken to reduce the negative effects, such as EcoRoute [Garmin 2013] and GreenGPS [Ganti et al. 2010; Saremi et al. 2015], where the detailed location information of traffic regulators is required to model cars' fuel consumptions when computing fuel-efficient routes. Therefore, the availability of road traffic regulator location information is crucial for this type of service as well. Furthermore, from the perspective of the development of future technology for the road, autonomous driving systems, such as Google's self-driving car [Google Self-Driving Car 2015], has begun to catch people's attention. It is not hard to imagine that having detailed traffic regulator information beforehand will help improve the robustness of such systems on the road. Despite their importance and potential benefits for the aforementioned fields, no nationwide database exists today that documents traffic light and stop sign locations (unlike the case with road maps). Instead, this information is quite fragmented, buried in physical archives of different counties and municipalities.

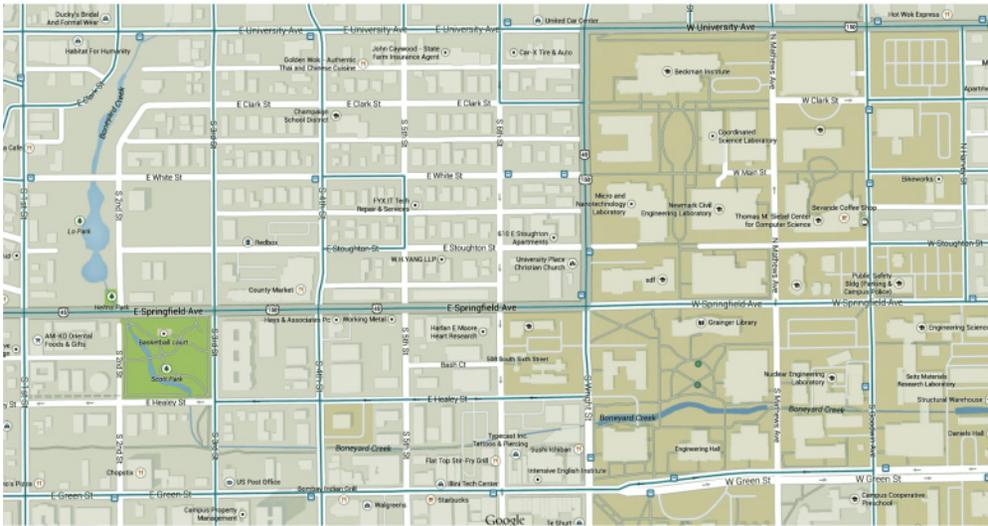
To address the preceding challenge, in this article we develop a novel crowd-sourced traffic regulator detection and identification system, called SmartRoad, that can automatically detect and identify stop signs and traffic lights from participatory sensing data shared by individuals from their vehicles. We investigate an approach that does not require explicit user action beyond downloading and running a new application on their smartphones. This goal eliminates techniques that are based, for example, on video recording or explicit human reporting, such as the types of data collection used for Google's sStreet View.

We would also like to point out that one of our important system design goals was easy large-scale deployment, which we think requires complete transparency to end users (requiring zero user effort for the setup and running, therefore, existing approaches like vision-based methods are not favorable because they require the phone be mounted in a way such that the phone camera can have a good view of the roads and signs ahead, which adds to additional end user efforts), and practical incentive strategies. We consider our current design choices to be well aligned with our goals, as our system can be simply piggybacked onto an mobile navigation app and immediately deployed to the mass; end users do not have to do anything different from how they normally use their phones for navigation for our system to function, and of course the detected traffic regulator information can then be used to enable and enhance additional features (e.g., green routing, driver assistance, etc.) of the navigation app, benefiting the users. Regarding the incentive strategy, end users who use our system and therefore share with us their driving data will be able to enjoy services that are enabled by our system; for example, green routing. This data-for-service strategy has already proven to work on existing mobile applications. For example, in the case of the Waze Social GPS Maps & Traffic mobile app (over 50 million downloads on Google Play store as of writing) [Waze 2014], users share their nearby road information such as accidents and jams, and in turn enjoy real-time traffic information, as well as several other benefits (e.g., location-based coupons, virtual community rankings, etc.).

Despite our general GPS-based system design as mentioned so far, an initial thought has been to write a script to crawl Google Street View itself (and use an appropriate visual analysis tool to extract stop sign and traffic light locations). However, as shown in Figure 1(a) (generated using the Google Maps [Google Maps 2014]), Street View is still only generally available in developed regions like the United States or Europe, and is virtually nonexistent for the most of the rest of the world. The coverage also gets



(a) General global coverage, shown as shaded regions on the map



(b) Per-road coverage around authors' department, shown as solid lines on the roads

Fig. 1. Google Maps Street View coverage.

spotty at times even in the United States; for example, if we zoom in the immediate neighborhood around the authors' department, as shown in Figure 1(b), we see many roads still do not have street-view data. Therefore, it is not a good idea to use street view for our purpose of designing a globally applicable solution. Thus, we focus on achieving accurate detection and identification performance using mobile phones.

Toward that end, a challenge that we need to address is the limitation on the collected raw data, embodied as data unreliability and sparsity. The unreliability comes from the general low quality of phone sensors, environmental noise, and/or improper handling of the phones, whereas the sparsity is a reflection of the fact that the amount of data for any particular single location is generally low despite the high amount of the entire set. In overcoming the data limitation, SmartRoad aggregates information along two dimensions. On one hand, it can dynamically adapt to application requirements and

data availability by selectively combining information acquired from multiple vehicles to reach a more accurate result. Here, the underlying philosophy is that different information sources, due to their inaccuracy and heterogeneity, usually observe the environment from different but complementary views. Therefore, aggregating the information of individual sources can often cancel out errors and reach a much more accurate result. This can be regarded as a major advantage of participatory sensing that advocates voluntary data collection and sharing from a large number of individual users. On the other hand, SmartRoad contains schemes that combine data traces at all the intersections to investigate the common vehicle moving patterns in the presence or absence of stop signs or traffic lights.

The other challenge that we face lies in the resource constraint, where resource is not only in the traditional system sense like network transmission energy, but also the availability ground-truth label information that is useful for our detection and recognition task (under the context of data mining and machine learning [Mitchell 1997; Han et al. 2011], and in this article, a label corresponds to the ground-truth information regarding the presence or absence of a stop sign or traffic light at a particular intersection). The SmartRoad system is designed to be flexible in adapting to dynamic resource constraints. On one hand, since users do pay real money for communication energy and cellular bandwidth usage, which are proportional to the amount of data that needs to be transmitted, it is desirable that sensors locally process the raw data they collect as opposed to forwarding them directly to the central server. To this end, SmartRoad can accordingly place the lightweight raw data processing operations on client phones. Information that is then sent to the server is of much smaller size compared to raw data, thus greatly reducing network transmission bandwidth requirements. On the other hand, ground-truth label information is usually limited (hence, systems like ours are meaningful in the first place). Our SmartRoad system is thus designed in a way that it can intelligently take advantage of any level of ground-truth label information, and whenever budget allows (e.g., hiring human labor to manually acquire more ground-truth information), actively evolve its core detection and identification engines to substantially improve performance.

The general design of SmartRoad follows a client-server framework. The client side consists of vehicle-resident smartphones, and the server runs on a back-end workstation. We have all witnessed the drastic growth in smartphone popularity in recent years, and several of their features make them an appealing platform for our SmartRoad system. (i) Smartphones today are packed with an array of onboard sensors on board, such as GPS, accelerometers, and barometers. (ii) Smartphone operating systems [Apple iOS 2014; Google Android 2013] allow users to install and run third-party applications that can potentially take advantage of the onboard sensors and provide services to the users, such as GPS-assisted local search and navigation. (iii) Compared with dedicated sensing, computing, or communication devices, smartphones are more suitable for large deployment due to their popularity. Users can easily download and install SmartRoad just like any other normal mobile applications. All of the preceding features make smartphones an ideal platform for our SmartRoad system.

We implemented SmartRoad on a vehicular smartphone test bed, and deployed it on 35 external volunteer users' vehicles for two months, collecting around 4,000 miles of driving data covering hundreds of regulator-controlled and uncontrolled locations. Experiment results show that SmartRoad can deliver outstanding detection and identification performance without consuming excessive communication energy/bandwidth or requiring too much ground-truth information.

The rest of this article is organized as follows. Section 2 provides a system overview; Section 3 discusses the details of the underlying detection and identification algorithms and schemes; implementation details are given in Section 4; experiment results and

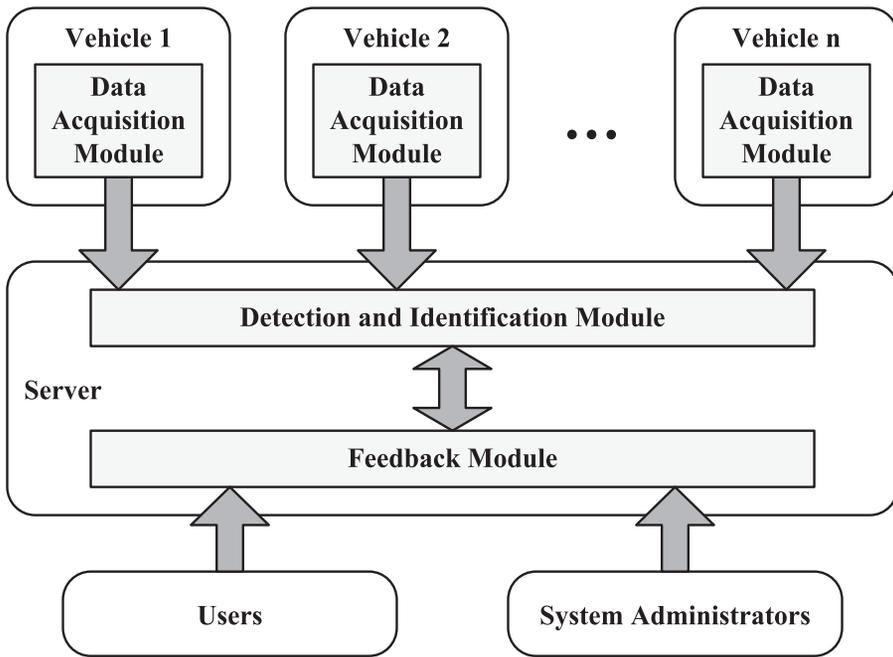


Fig. 2. Architecture design overview of the SmartRoad system.

evaluation are presented in Section 5; Section 7 talks about various related work; and finally Section 8 concludes.

2. SYSTEM OVERVIEW

In this section, we provide an overview of SmartRoad, briefly touching upon the various system components, namely, the data acquisition module, the detection and identification module, and the feedback module, as illustrated in Figure 2.

The data acquisition module runs on vehicle-resident smartphones. It performs the following three major tasks. (i) It accesses the various onboard sensors of the phone to collect sensory data of the physical environment. (ii) It processes the raw data collected from the sensors according to the various data representation and transmission schemes for the detection and identification algorithms. For any specified application performance requirement, the data acquisition module minimizes the amount of data that needs to be delivered to the central server in order to save network transmission energy and bandwidth. (iii) It uploads the processed data to the central server. Cellular data networks and open public WiFi networks are both valid options as transmission channels. However, they each have their own limitations such as being potentially monetarily costly (in the case of cellular) or being generally of poor coverage (WiFi). SmartRoad gives users the choice to use either or both communication channels based on their own personal situations and preferences.

These three tasks are all potentially battery hungry operations. Therefore, a standalone energy management daemon is also run on the phone, to monitor the battery/charging status of the phone and manage the execution and halting of the power-hungry operations. This is to ensure that our client system has minimal negative effect on the other phone functionalities/operations or the general user experience.

The detection and identification module runs on the central server, and processes data received from vehicle-resident phones. Thus, its role is twofold. On one hand, it behaves like a data server, listening for incoming connections and receiving data

from vehicle-resident phones, for which data can be either directly piped to the final processing engine, or stored in a database for later batch processing. On the other hand, it serves as a central data processor, as it carries out detection and identification tasks, that is, makes decisions for each intersection regarding the presence/absence and the type of traffic regulators. The actual detection and identification algorithms are discussed in detail in Section 3.

Note that the exact operation carried out in the central processor is dynamically tuned to cope with the level of data processing done on the phones as dictated by the specific application performance requirement. For example, the central processor would carry out different sets of operations when it receives, from the phones, data that has already been somewhat processed than if the phones directly send raw sensory data to the central server without carrying out any local data processing at all.

The feedback module also resides on the central server. As illustrated in Figure 2, the information flow between the detection module and the feedback module is bidirectional. On one hand, it takes as input the detection results and presents them to next tier applications that take as input the detection and identification results (e.g., assisted driving or navigation systems such as GreenGPS [Ganti et al. 2010; Saremi et al. 2015], or simple visualization of the detection and identification results via a web service interface). On the other hand, the web service interface can be used by system administrators and authenticated users to correct detection errors and provide ground-truth information, which is then piped back to the detection and identification module for dynamic and adaptive performance improvement, as discussed in detail in Section 3.

3. DETECTION AND IDENTIFICATION

We now present in detail the algorithms that we developed for the traffic regulator detection and identification tasks.

In tackling this problem, the most intuitive approach would be to use simple heuristic-based rules, which is essentially what the state-of-the-art method [Wang et al. 2013a] does, where, for example, rules are set to recognize a location as having a traffic light if a vehicle waits there for 15–90sec, and a stop sign 2–10sec. However, the problem with these heuristic-based rules is that the manually selected decision points (e.g., 15–90sec, 2–10sec) might only be applicable to certain subregions of the problem space but not some others—it is quite possible that the rules as mentioned previously work quite well for a rural area, but fail miserably under an urban setting. Moreover, we can potentially make multiple rules for the same recognition task, in which case, it is not clear how the manual heuristic methods can weight the multiple rules and possibly combine them, which, otherwise, might be able to improve the recognition performance. Therefore, in designing our SmartRoad system, how decision points are to be set and how the rules are to be weighted and combined together need be carried out in a systematic and automatic fashion.

Toward this goal, we adopt statistical classification techniques [Mitchell 1997] as the core detection and identification components for our SmartRoad system. We design a set of skeleton rules that characterize various aspects of people’s driving patterns around road intersections. Given actual runtime data collected from the target problem domain, it is entirely up to the system to automatically explore the problem space and set internal parameters accordingly. Compared to the heuristic method mentioned previously, our SmartRoad system is able to not only optimally select the most suitable decision points, but also automatically carry out the most effective rule selection by placing weights and deciding their combination patterns for use in the detection and identification tasks. Experiment results, as later shown in Section 5, confirm the advantage of our approach over the simple heuristic method.

For the rest of this article, Sections 3.1 and 3.2 describe the client-side raw data preprocessing and preparation; Sections 3.3 and 3.4 present the details for the server-side detection/identification components and techniques; and in Section 3.5, we discuss the strategies that we employ to enable SmartRoad to be able to meet different application transmission/performance requirements when carrying out its sensing tasks.

3.1. Data Preprocessing

The raw GPS trajectory data undergoes several preprocessing steps before the actual classification operations are carried out, as explained in the following.

First, the raw GPS trace data are broken into separate traces by dividing between consecutive GPS frames where the time-stamp difference exceeds a threshold (corresponding to users parking, or powering off cars for the night, etc.). Each trace then represents a separate continuous driving activity, and the trace data will only reflect the physical driving process rather than also containing stationary artifacts caused by non-driving-related actions (e.g., parking). Whenever the client phone-side data acquisition module detects the end of a trace, all preprocessing and subsequent feature extraction operations (as will be discussed shortly) are carried out for that trace.

Each trace is then scanned through and broken into segments by using road intersection location information. Each segment represents the GPS trace portion that leaves some immediate previous intersection and approaches and arrives at a current intersection. Therefore, each intersection visited by a GPS trace will have one or more associated segments. The intersection location information needed can be easily extracted from the Open Street Map [2014] data, and it is the only map information that we use in our entire system.

As a side note, trace segments involving cars making turns can potentially confuse a detection and identification system, since the resulting approach→stop→turn movement patterns observed at uncontrolled intersections are essentially indistinguishable from that of stop sign controlled ones. Thus, for each intersection, we need to identify and remove the crossing segments that involve cars making turns. To achieve this, we simply measure, for each intersection, the car bearing (driving direction) on the approaching segment and that of the leaving segment. Turnings are therefore indicated by considerable bearing differences, say greater than 45° , between the two. Straight crossing segments are therefore preserved.

Please note that special attention needs to be paid to *T*-shape intersections. Let us denote the left, right, and bottom ends of a *T*-shape intersection *A*, *B*, and *C*, respectively. The traffic coming from *C* into the intersection (let us call this type of traffic *C*-traffic for ease of discussion) must always turn, and thus will always be discarded. However, in our ground-truth manual collection as described later in Section 5, we observed that all *C*-traffics are always controlled, and *A*- and *B*-traffics always share the same regulator types. Furthermore, if *A/B*-traffic is uncontrolled, *C*-traffic is stop sign controlled; otherwise, *C*-traffic has the same control type as *A/B*-traffic. In other words, if *A/B* has traffic lights, so does *C*; otherwise *C* has stop sign. Therefore, by discarding turning traffics in our system design, we are not missing the *C*-traffic regulator types of *T*-shape intersections.

Lastly, we decompose each intersection location into multiple data instances, as intersections generally have three or more incident approaching directions, which might or might not have different traffic regulator situations. For example, a four-way intersection might only have stop signs installed for the north-south direction, leaving the east-west direction uncontrolled. We therefore decompose every single intersection into multiple ones, each represented as a <latitude, longitude, direction> tuple. Hence, the four-way intersection in our previous example would be decomposed to four different ones, and the traffic regulator status of each of them is considered separately.

3.2. Feature Extraction

Given the preprocessed data, the next step is to design a set of rules that can characterize the different driving patterns generally displayed and observed for the differently regulator-typed intersections. In the context of statistical classification, this is equivalent to feature extracting from the GPS data. We extract five features, all of which are inspired by everyday common sense (some are also partially used by prior work [Carisi et al. 2011]). Next, we give detailed descriptions of the intuition and extraction for all five features.

- (I) **Final Stop Duration:** This feature captures the time duration of the last stop that a car makes in front of an intersection before crossing. The intuition is: At stop signs, cars generally all stop, but perhaps only for a few seconds; at uncontrolled intersections, however, cars probably do not stop and thus the stop duration will be 0 or at least quite small; for red lights, all cars would stop, and the durations are possibly longer than that of stop signs. The actual extraction involves simply scanning through the segment data and checking the time duration of the last continuous 0-speed block, if any.
- (II) **Minimum Crossing Speed:** This feature represents the lowest speed at which a car crosses an intersection. It should be quite low for stop signs, and relatively high for uncontrolled intersections; for traffic lights, a mixture of low and high values might be observed, depending on the actual light conditions. To extract this feature, we scan the second half of each segment and pick the lowest speed reading.
- (III) **Number of Decelerations:** This feature captures the number of times that a car decelerates as it approaches an intersection. For uncontrolled intersections and green lights, cars generally do not decelerate; for stop signs and red lights, one or more decelerations might be observed. To extract this feature, we use speed difference and time duration thresholds to define a continuous deceleration block in segment data, and then just count the number of such blocks.
- (IV) **Number of Stops:** This feature captures the number of stopping actions in a segment. For uncontrolled intersection and green lights, no stopping is expected; for red lights, one is expected, regardless of whether the car is right in front of the intersection or queued behind several cars; for stop signs, one is expected if the car approaches the intersection with no other cars in front of it, and multiple if the car is queued behind several cars in front of a stop sign, as each of the multiple cars needs to come to a full stop at the stop sign before proceeding, causing following cars to likely also display multiple rounds of stop-and-go motions. To extract this feature, we simply count the number of continuous 0-speed blocks in the segment.
- (V) **Distance from Intersection:** This feature measures the distance between the intersection coordinate location and the point where a car makes its last stop, if any, before crossing the intersection. If a car does not stop before crossing an intersection, we mark this distance as -1 . The intuition here is as follows. For uncontrolled intersections and green lights, this value should be -1 ; for stop signs, this distance should be quite small as all cars are expected to stop right under the stop signs in front of the intersections; for red lights, this distance could be relatively larger as a car might get queued behind multiple cars. To extract this feature, we identify the final 0-speed block, if any, in the segment and measure its distance from the intersection point.

Note that all preprocessing and feature extractions are done locally on the vehicle-resident smartphones in real time. As they are all extremely lightweight operations, the computation energy consumption is negligible.

Our system extracts the five aforementioned *physical features* for all GPS segments corresponding to crossings through intersections. All the per-segment physical feature vectors corresponding to the same intersection are then gathered together, from which certain statistics (e.g., mean and variance) are computed for each of the five physical dimensions. In this article, we refer to these statistics as *statistical features*, and feed them as the immediate input to our detection and identification engine, which in turn classifies each intersection as having traffic light, stop sign, or simply being uncontrolled.

In SmartRoad, we design and implement two different classification modes, namely, *supervised classification* mode and *unsupervised classification* mode, to carry out the actual detection and identification tasks under different application settings. As suggested by the names, their applicability depends on the availability of training dataset, in which each data point is associated with a ground-truth class label. Given a set of training data, supervised classification methods are appealing because label information can help capture the inherent structure of the target data automatically. Without label information, unsupervised models can partition the data only based on the intrinsic similarities among data points corresponding to the same ground-truth label. In this article, we want to design our system in a way that it is able to handle any availability level of label information, and does so in a dynamic and intelligent manner. Thus, in the next two subsections, we provide detailed explanation and discussion of these two classification components.

3.3. Supervised Classification

In the dataset collected and preprocessed as previously discussed, each data point is a vector composed of the statistical features of a particular intersection. A training set in this context corresponds to a group of intersections with their corresponding ground-truth traffic regulator information. In practice, ground-truth label information is limited and expensive to acquire, thus we consider a realistic scenario where initially only a tiny amount of training data is available, with further label information being acquired incrementally either on demand or opportunistically.

Regarding the actual classification technique, we use random forest [Breiman 2001] as our base supervised classifier. Random forest is a decision tree based classification algorithm that trains multiple decision trees simultaneously and has them vote for the final classification decision. The training of decision trees is much like the derivation of rules, but in an automatic and systematic manner. Provided with labeled training data, each decision tree iteratively figures out the optimal decision features and boundary values (branching points) according to the labeled training data. For the selection of the base classification technique, we experimented with a set of widely used algorithms (Gaussian Mixture Models, Support Vector Machines, and Naive Bayes) besides Random Forests and picked the one that yielded the highest classification accuracy. We wanted to focus our work more toward the designing of a general system framework that can make the best use of any classifier incorporated. Therefore, we designed our system such that more advanced and sophisticated classification algorithms can easily be plugged in as the base classification core with the rest of the system components intact.

As the core component of our SmartRoad participatory sensing system, the supervised classification engine needs to be able to dynamically adapt to various application requirements and evolve to fit different sensing application conditions. In particular, we design and implement two adaptive mechanisms in our supervised classification engine, as we discuss in detail in the following.

3.3.1. Active Learning Adapter. In realistic sensing application scenarios today, it is feasible that budget allows one to manually acquire ground-truth information (e.g., Google's

ALGORITHM 1: Active Learning Adapter

Input: The set containing all data that need to be classified $\mathcal{D}_{\text{target}}$; supervised classifier $\mathcal{C}_{\text{classify}}$; labeled training dataset $\mathcal{D}_{\text{train}}$; high confidence threshold h_{conf} ; low confidence threshold l_{conf}
Output: High confidence label set $\mathcal{L}_{\text{conf}}$ that holds all final classification results

```

1: Initialize empty set  $\mathcal{L}_{\text{conf}}$ 
2: while  $|\mathcal{D}_{\text{target}}| > 0$  do
3:   train  $\mathcal{C}_{\text{classify}}$  on  $\mathcal{D}_{\text{train}}$ 
4:   apply  $\mathcal{C}_{\text{classify}}$  on  $\mathcal{D}_{\text{target}}$  to get result labels  $\mathcal{L}$  and the corresponding confidence scores  $S$ 
5:   for  $i \leftarrow 0$  to  $|\mathcal{D}_{\text{target}}|$  do
6:     if  $S[i] \geq h_{\text{conf}}$  then
7:        $\mathcal{L}_{\text{conf}} \leftarrow \mathcal{L}_{\text{conf}} \cup \{\mathcal{L}[i]\}$ 
8:        $\mathcal{D}_{\text{target}} \leftarrow \mathcal{D}_{\text{target}} \setminus \{\mathcal{D}_{\text{target}}[i]\}$ 
9:     else if  $S[i] \leq l_{\text{conf}}$  then
10:      manually label  $\mathcal{D}_{\text{target}}[i]$ 
11:       $\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_{\text{train}} \cup \{\mathcal{D}_{\text{target}}[i]\}$ 
12:       $\mathcal{D}_{\text{target}} \leftarrow \mathcal{D}_{\text{target}} \setminus \{\mathcal{D}_{\text{target}}[i]\}$ 
13:     end if
14:   end for
15: end while
16: return  $\mathcal{L}_{\text{conf}}$ 

```

manual collection effort of acquiring Street View images). However, we need to assume that this budget is usually limited to support the manual collection of only up to some small amount of ground-truth information compared to the size of the entire sensing task. Thus, an important question to ask here would be: *For which intersections should we pay to manually acquire their ground-truth information in order to maximize our system detection and identification performance?* The most natural approach here would be to just randomly select intersections up to the amount permitted by the budget. Though straightforward and easy to carry out, this scheme does not necessarily lead to the optimal utility of the limited budget.

A more intelligent approach here would be to look at the past classification results, identify the intersections for which the classification algorithms are the least confident about, and then hire people to manually acquire the ground-truth information for those particular (*difficult*) intersections. Here we borrow the active learning philosophy [Dasgupta and Langford 2009; Krause and Guestrin 2007]. We call this mechanism the *Active Learning Adapter* for our supervised classification engine, the detailed steps of which are shown in Algorithm 1. In particular, our system loosely computes the confidence score as the ratio of the winning vote among all votes from the random forest's bag of decision trees. For example, when classifying an intersection, if the trees of a 10-tree random forest cast seven votes for traffic lights and three for uncontrolled, then the final classification result label will be traffic lights, with a confidence score of 0.7.

3.3.2. Self-Training Adapter. Unlike the active learning adapter, which focuses on strategically acquiring new ground-truth labels, the *Self-Training Adapter*, which adopts the idea of self-training [Zhu 2008], looks at its own past classification results and tries to take advantage of them to improve system performance. More specifically, the classified intersections that have the highest confidence scores from the classifiers are progressively collected and added to the training set. The intuition behind this is that classification results with high classification confidences are most likely to be correct, and thus considering the inferred label information of these data points as ground truth increases the size of the training set without needing to spend any of the limited budget, the savings of which can then be spent more toward other more meaningful

ALGORITHM 2: Self-Training Adapter

Input: The set containing all data that need to be classified $\mathcal{D}_{\text{target}}$; supervised classifier $\mathcal{C}_{\text{classify}}$; labeled training dataset $\mathcal{D}_{\text{train}}$; high confidence threshold h_{conf}

Output: High confidence label set $\mathcal{L}_{\text{conf}}$ that holds all final classification results

```

1: Initialize empty dataset  $\mathcal{L}_{\text{conf}}$ 
2: while  $|\mathcal{D}_{\text{target}}| > 0$  do
3:   train  $\mathcal{C}_{\text{classify}}$  on  $\mathcal{D}_{\text{train}} \cup \mathcal{L}_{\text{conf}}$ 
4:   apply  $\mathcal{C}_{\text{classify}}$  on  $\mathcal{D}_{\text{target}}$  to get result labels  $\mathcal{L}$  and the corresponding confidence scores  $S$ 
5:   for  $i \leftarrow 1$  to  $|\mathcal{D}_{\text{target}}|$  do
6:     if  $\mathcal{L}_{\text{score}}[i] \geq h_{\text{conf}}$  then
7:        $\mathcal{L}_{\text{conf}} \leftarrow \mathcal{L}_{\text{conf}} \cup \{\mathcal{L}[i]\};$ 
8:        $\mathcal{D}_{\text{target}} \leftarrow \mathcal{D}_{\text{target}} \setminus \{\mathcal{D}_{\text{target}}[i]\};$ 
9:     end if
10:  end for
11: end while
12: return  $\mathcal{L}_{\text{conf}}$ 

```

ground-truth acquisition targets as explained previously in Section 3.3.1. Algorithm 2 demonstrates how our self-training adapter works.

3.4. Unsupervised Classification

Toward designing a robust detection and recognition system, we also need to consider the though unlikely but possible extreme deployment condition of having no ground-truth information available, for which case the previously discussed supervised classification methods would not work. In addressing this problem, we also developed an unsupervised classification engine for SmartRoad. Unsupervised classification (or clustering) algorithms generally group data based on their internal similarity instead of using label information. Compared with its supervised counterpart, unsupervised classification methods are more application independent and easily deployable, due to their lack of requirements for training data.

For our SmartRoad system, we use spectral clustering [Ng et al. 2002] as the base unsupervised clustering algorithm. As opposed to traditional clustering algorithms (e.g., k -means or generative expectation-maximization (EM) framework) that always result in convex sets, spectral clustering can solve problems in much more complex scenarios since it does not make assumptions on the form of the cluster or the distribution of the data. We carefully study the data used as the input to the classification component, and find that they neither display convexity nor follow mixture models. Our experiments also show clear advantages of spectral clustering over traditional clustering algorithms on our data.

To incorporate clustering methods, there is one question that needs to be answered: how to infer the class labels for data clusters. Our system achieves this through investigating the feature statistics of the data within different clusters. In Section 5, we demonstrate the label derivation for the clusters discovered by our unsupervised classification engine.

3.5. Implementation Strategies

Different from traditional classification tasks that assume a central database, classification for participatory sensing applications is faced with a major challenge, namely, the sensory data are distributed over a large amount of participating users. Generally, there is a natural trade-off between the amount of information delivered by each user and the classification performance. More accurate results can usually be achieved when a larger amount of information is transmitted, which, however, might lead to higher

bandwidth consumptions. For the traffic regulator detection and identification tasks, we substantiate this trade-off into two dimensions of system design choices: (i) For each vehicle, what features should be extracted from the data and submitted to the central server, and (ii) what level of information aggregation needs to be done in the server. The combination of these two choices determines the amount of information delivered by each vehicle, and has an immediate impact on the detection and identification performance. We now discuss each of these two dimensions in more detail.

3.5.1. Information Aggregation. As previously discussed, we extract from raw data five different physical features, and derive statistical features from each of them. The resulting statistical features associated with each intersection are used as input fed to the classification engines to infer the regulator type. In SmartRoad system, the extraction of physical features always takes place locally on each phone. No raw GPS data ever needs to be transmitted. Depending on where and how the derivation of statistical features takes place, we design three different information aggregation schemes, each of which corresponds to a different level of data abstraction.

- (I) **Data Aggregation (DA):** The original physical feature vector extracted for every crossing through an intersection is transmitted to the central server. The server collects all physical feature vectors corresponding to the same intersection and computes a statistical feature vector for that intersection.
- (II) **Feature Aggregation (FA):** The per-intersection statistical features are computed by each individual user, and are then opportunistically sent to the server, along with the associated crossing segment count information. All the information from different users is then used to estimate the final statistical features by computing the weighted average using users' segment counts for each intersection. Therefore, FA shrinks DA's per-crossing segment transmission level down, by one order of magnitude, to per intersection.
- (III) **Label Aggregation (LA):** Each individual user carries out classification locally by phone and only sends the classification result labels to the server, who then derives the final labels by carrying out weighted voting, using the same count information mentioned in FA. Compared to FA, this scheme further shrinks the amount of data that needs to be transmitted for each intersection down from a feature vector to essentially a single local decision label.

3.5.2. Feature Selection. To thoroughly explore the transmission-performance trade-off space, we design three sets of statistical feature selection schemes, as discussed next.

- (I) **Default:** The default statistical feature extraction scheme consists of computing the minimum, maximum, mean, and variance (min-max-mean-var) on sets of physical features. This scheme captures a straightforward statistical signature associated with every intersection as users drive around and the number of crossings at individual intersections accumulates. Under this scheme, each intersection is associated with a $4 \times 5 = 20$ dimensional statistical feature vector.
- (II) **All Spectrum (a-):** For the sole purpose of capturing a richer statistical signature than the default min-max-mean-var, we also design the all-spectrum statistical feature extraction scheme, which replaces the default min and max components with percentile values ranging from 0 to 100 with 5 as the stepping length. Therefore, under the all-spectrum scheme, every intersection is associated with a $\{[(100-0)/5+1]+2\} \times 5 = 115$ dimensional statistical feature vector. We expect this scheme to yield relatively higher classification performance even though it is quite apparent that it would incur a heavier burden for the transmission channel than the default scheme.



Fig. 3. The data acquisition module running on a Galaxy Nexus smartphone, deployed in car.

- (III) **Economic (e-):** For this scheme we simply drop the variance value from the default scheme. At first glance, it might seem like a minor modification, the transmission requirement implication for the DA information aggregation scheme, however, is rather significant. The DA scheme, as just introduced, needs to send per-crossing segment information to the server; under the modified scheme, eDA, however, a user needs only send the per-intersection min-sum-max values, along with the per-intersection crossing counts, to the server, where the global per-intersection min-mean-max statistics can be recovered. Therefore, eDA has a transmission profile one order of magnitude smaller than the original DA scheme.

Combining the information aggregation and feature selection schemes, we have $3 \times 3 = 9$ classification scenarios in total, the performance analysis of all of which is presented in Section 5. Note that for the three label aggregation schemes, namely, LA, eLA, and aLA, classifications are carried out on phones locally, whereas for all other schemes classifiers run on the central server.

4. SYSTEM IMPLEMENTATION

In this section, we talk about the implementation details of the various component modules of our system. An in-vehicle deployed system is illustrated in Figure 3.

4.1. Data Acquisition Module

The vehicle-resident smartphone-based data acquisition module has four main components, implemented to carry out different functionalities, as follows.

Data Collection: The data acquisition module is implemented on Google's Galaxy Nexus Android phones, equipped with 1.2GHz dual core CPU, 1GB memory, and 16GB

flash storage, running on Android 4.0 operating system [Galaxy Nexus 2013]. We collect readings from the following phone sensors: (i) *GPS Sensor*, the main source of the data to be used for our detection and identification tasks. Every single GPS reading includes the instantaneous latitude-longitude location, speed, and bearing of the vehicle; and (ii) *Power Sensor*, which is used to determine the phone charging status, which reflects the car's engine on/off status, which is then used by the phone side code to start or stop data collection and communication.

Data Processing: The data processing component is responsible for carrying out physical and statistical feature extractions according to the appropriate representation and transmission schemes. Note that even though phones are becoming more and more computationally capable, they are still far less powerful compared to traditional workstations; Attempting to carry out data processing tasks that are too computationally intensive can potentially affect the running of other applications and affect the general user experience. However, as discussed in Section 3, the computation requirements for the feature extractions involved in our detection and identification tasks are all quite lightweight and thus are all done easily on phones.

Data Delivery: A traditional client-to-server uploading module resides on each phone and opportunistically transmits data back to the central server upon the availability of some appropriate communication channel. It is, however, also possible that some users never drive within public WiFi areas, nor do they choose cellular data network as a usable channel. We therefore also implement, on phones, a peer sharing module that lets any two phones potentially share their data with each other, thus increasing the possibility of the data reaching the central server sooner. The peer sharing module amends the traditional client-to-server uploading module and forms a delay-tolerant network (DTN) for pushing nodes' data to the server.

Energy Management: Pulling GPS or carrying out various network communication tasks are all energy hungry operations. Due to the nature of the deployment environment, users can choose to plug their phones into the car chargers. The SmartRoad energy management component checks the phone's battery level and charging status and suspends/revives the power-hungry operations accordingly. This also helps achieving completely autonomous phone operations during our experiments.

4.2. Detection and Classification Module

The detection and classification module is implemented on a back-end central server, handling the following two main tasks.

Data Server: The multithreaded data server is written in Java, capable of handling multiple TCP connections from clients. The server assigns one worker thread per incoming TCP connection. Each worker thread reads data from the input stream and converts them into a usable format on the fly. Before the conversion, parameter text labels are encoded in a static numeric representation to shrink data transfer size. After the conversion, the server performs a lookup on the numeric values before generating SQL insert statements. The table in the database uses an auto increment field as the primary key for simplicity, and are indexed by the tuple (PhoneID, SampleID) to reduce data insertion overhead. The indexing enables MySQL to organize the stored data into B-tree data structures to speed up duplication detection when inserting new data.

Central Processor: The core component for carrying out the actual traffic regulator detection and identification tasks is implemented in Python, which enables straightforward integration with the data server and the feedback module.

4.3. Feedback Module

The feedback web interface is built using the Django [2014] framework. It provides a graphical user interface for easy interactions. It uses the Python plugin [Celery 2014] to periodically update the traffic regulator detection and identification results as well as

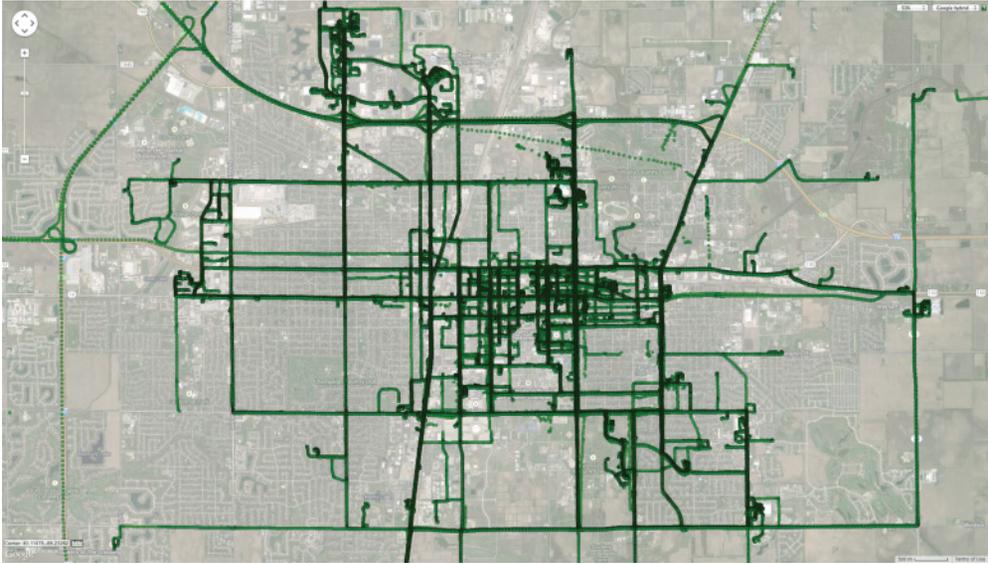


Fig. 4. Google Maps overlay of the GPS traces collected. For ease of illustration, the map is centered and focused around the Urbana-Champaign IL area, where most of the data collection took place.

data plots and driving statistics so that the user can retrieve timely data. Additionally, users and system administrators can interact with the interface to correct mistakes and/or insert ground-truth information.

5. EXPERIMENTS AND EVALUATION

In this section, we describe our experiments, and present the performance evaluation on various aspects of our SmartRoad system.

We recruited 35 external volunteer participants for our study, and deployed our SmartRoad system client phones in their vehicles. All the phones operate in a completely autonomous manner, requiring no user intervention from our participants. All users were asked to drive just as how they normally do. The entire experiment lasted for about two months with each individual user averaging about three weeks' duration. Roughly 4,000 miles of driving data were collected, covering a total of 158 traffic lights, 77 stop signs, and 228 uncontrolled unique intersections. An illustration overlaying Google Maps of the entire GPS traces collected is shown in Figure 4. In Figure 5, we show the distribution of the three different intersection regulator types from the driving data collected, with key percentile (25th, 50th, and 75th) statistics summarized in Table I. In order to have accurate ground-truth information for evaluation purposes, we made several attempts to acquire such information from our municipalities but received no useful response, unfortunately. Thus, spending a huge amount of time and effort, we *physically* went to *all* road intersections in our Urbana-Champaign main data collection area and *manually* recorded the corresponding latitude/longitude/direction and traffic regulator information.

Our various experiment results show that SmartRoad can effectively and efficiently execute its participatory sensing and classification tasks under various application requirements and conditions with high accuracies and small transmission requirements. About 80% accuracy can be achieved without any ground-truth label information or handcrafted commonsense-based rules; more than 90% accuracy is reached with less than 20% ground-truth training information. Under realistic sensing application

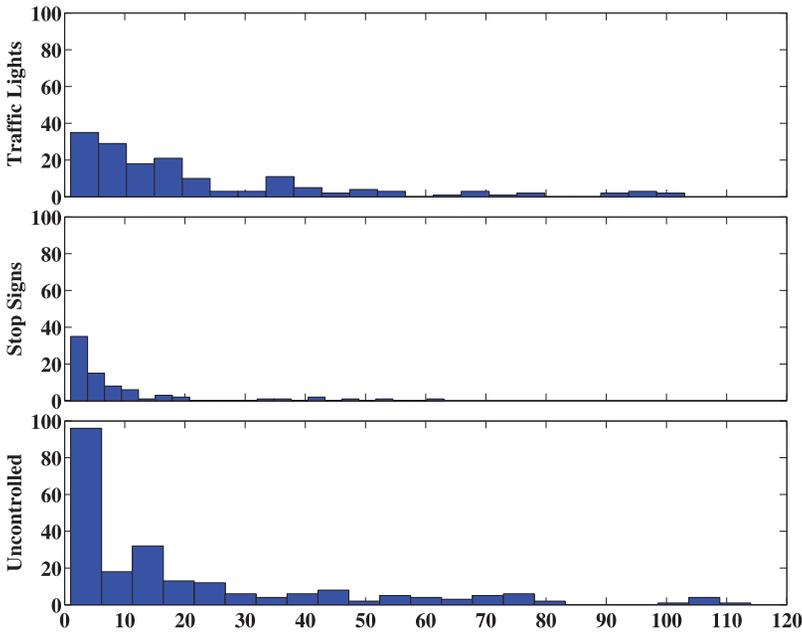


Fig. 5. Intersection traffic regulator type distribution (histogram) of our collected driving data. The x -axis indicates the number of times an intersection was driven past. For example, the leftmost bar in the bottom plot says that there are 96 uncontrolled intersections where each one of them was driven past five times or less.

Table I. Intersection Type Statistics

Percentiles	25th	50th	75th
Traffic lights	6	14	34
Stop signs	1	4	11
Uncontrolled	2	10	26

The percentile statistics for the various intersection regulator types of our collected driving data.

scenarios, SmartRoad claims and maintains up to 10% accuracy advantage margin over the baseline approach at all manual ground-truth acquisition budget levels, which also translates to more than 50% savings in terms of budget required to achieve the same detection accuracy.

We next present the detailed experiment results.

5.1. Supervised Classification

5.1.1. Evaluating Information Aggregation Schemes. First, we evaluate the classification performance for the three information aggregation schemes, namely, DA, FA, and LA, as defined in Section 3.5.1.

Note that for the sake of presentation clarity, here we only show the average results for all three classes. The actual underlying per class results are indeed balanced, without any particular class being favored or sacrificed in the classification, as demonstrated in a later evaluation segment.

Generally, as the amount of data, or labeled training data, or both, increases, classification performance also improves, as clearly shown in Figure 6. It can be observed from these figures that for all data amounts or training data availability situations,

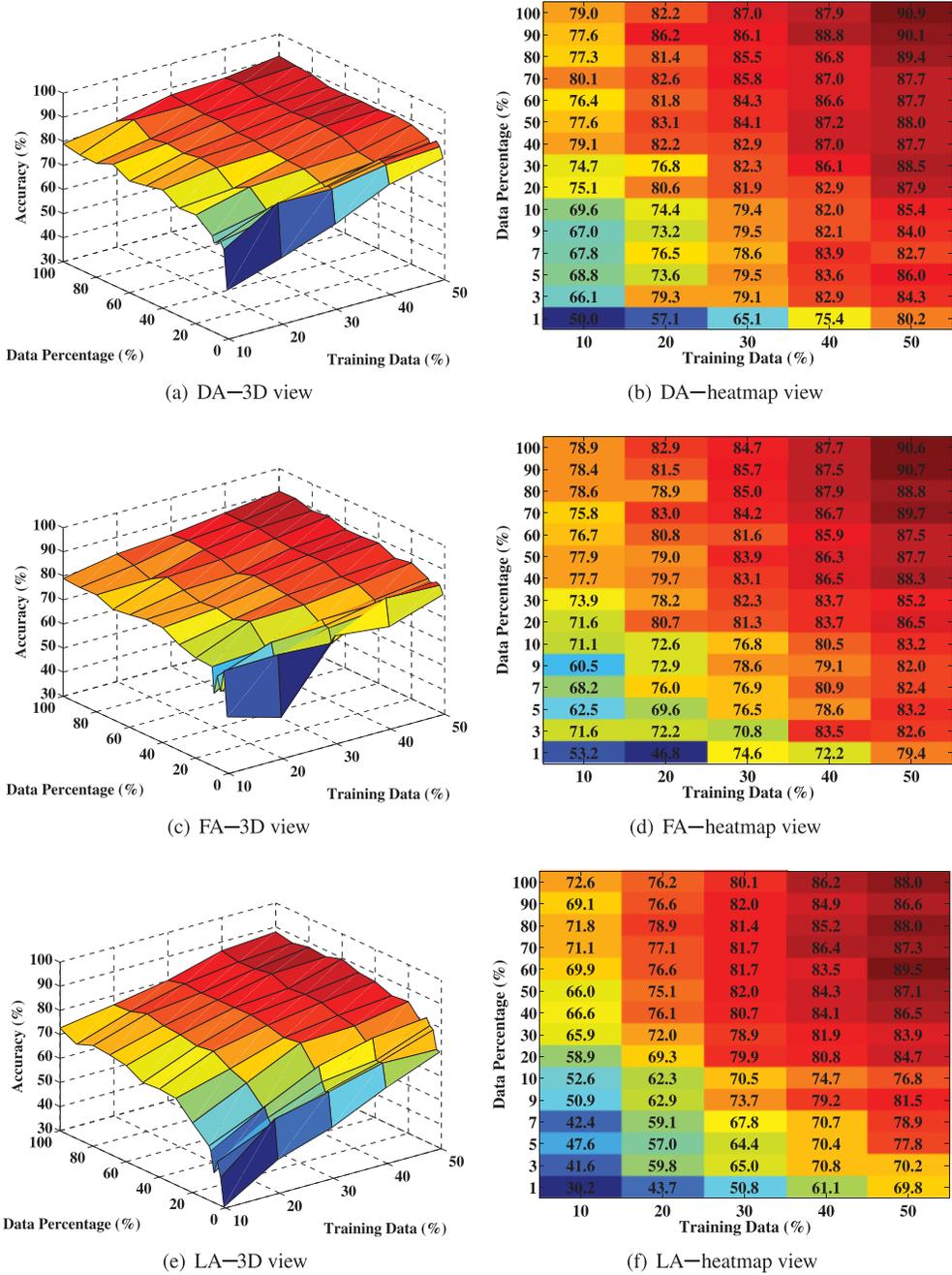


Fig. 6. Supervised classification performance

the classification performance generally follows the DA>FA>LA trend, which agrees with the network transmission requirement for each of them. At system start up, with extreme data limitation in general, all three schemes perform quite poorly, with LA being the worst of all. As the amount of data and/or labeled training data availability increases, the differences among all three schemes diminish. This suggests that

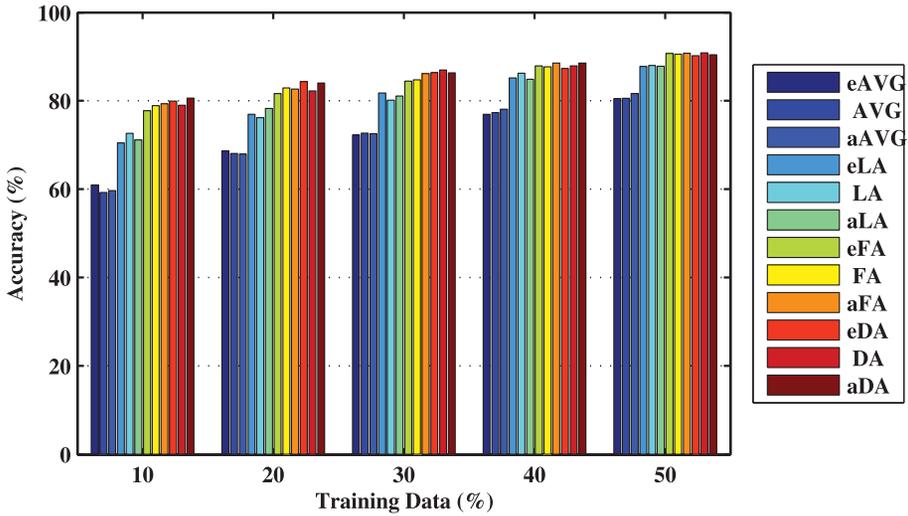


Fig. 7. Classification accuracies at varying training data availabilities.

as the deployment of a participatory sensing system elapses and data flows in and accumulates, feature selection and information aggregation schemes can potentially be adaptively modified to reduce the communication requirements without incurring much damage to the application performance.

5.1.2. Evaluating Feature Selection Schemes. We next take a look at how different feature selection schemes compare with each other under various data and training set availability settings. We experiment with different physical feature selection schemes, and discover that using all five always yields the best performance. Thus, here we only evaluate the different statistical feature schemes, which also have far more impact on the resulting transmission requirements. Notice from the previous set of experiment results shown in Figure 6 that the amount of data has a smaller effect on classification performance than training data availability does. Therefore, for the next set of experiments, we fix the total dataset at its full size and examine the classification performance at varying training data availabilities and feature selection/information aggregation schemes, as Figure 7 illustrates.

Note that the included AVG is not an information aggregation scheme per se. Instead, it refers to the classification scheme where each user uses his/her own local data to classify the intersections he/she ever passes through. This helps demonstrate both qualitatively and quantitatively how crowd-sourced participatory sensing can help achieve application performance that is otherwise hard to imagine for single users.

As shown, the trend that more training data means higher performance holds true for all scheme combinations, as well as for individual users. However, it can be observed that individual users, when acting alone, do not quite catch up with the various aggregation schemes even when the amount of training data becomes abundant, illustrated by the lagging behind of the leftmost three bars at all training data availability levels. We also take a closer look at the performance measures at 10% training data availability, which is a quite harsh condition for supervised classification methods. We see that individual users have their accuracies averaged at around 60%. However, simple voting among the users would immediately improve the accuracies to over 70%; and any slightly more sophisticated scheme would further bump the accuracies to around

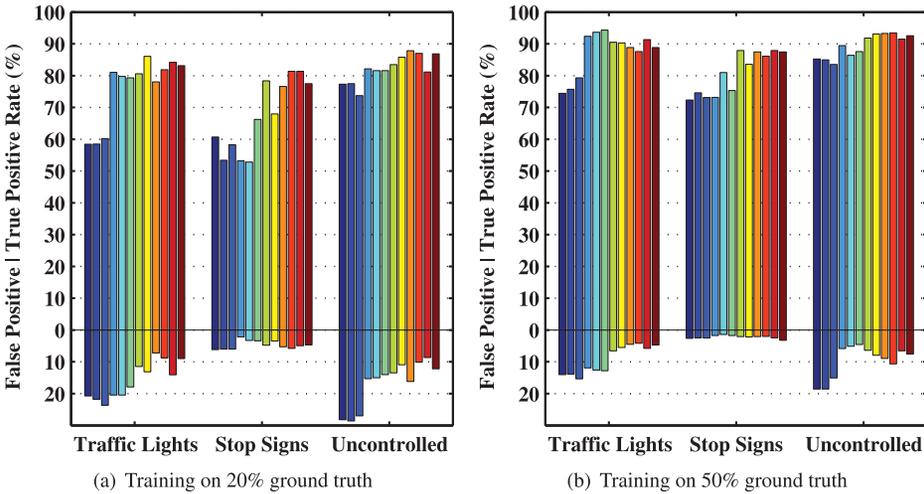


Fig. 8. The per-class classification performance. The different colors indicate the different aggregation schemes we experimented with; the exact color-to-scheme correspondence is the same as that of Figure 7. For every single aggregation scheme, the upward bar (above $y = 0$) indicates the classification's True Positive rate, and the downward bar (below $y = 0$) the False Positive rate. For example, with 20% training data (Plot (a)), traffic lights classification (leftmost cluster) under the eDA scheme (the third bar from the right) yields about 83% true positive rate (upward portion of the bar) and 9% false positive rate (downward portion).

80%. This is a clear demonstration of the power of crowd-sourced participatory sensing, under considerable label constraints.

5.1.3. Individual Class Classification Performance. So far, we have been looking at classification results by means of weighted average among all three target classes, namely, traffic light controlled, stop sign controlled, and uncontrolled intersections. Next we take a closer look at the individual class classification results. We present two sets of experiment results with different levels of training data availabilities, as shown in Figure 8. We have the following two observations: (i) the average classification performance for both training data availability levels agrees with the results shown in Figure 7, and (ii) even though certain per class performance differences can be observed, it still is evident that SmartRoad classification engines treat all three target classes fairly, without favoring or sacrificing any particular classes. Therefore, for the sake of presentation clarity, we do not show any more per-class classification results.

Before moving on, we compare our per-class classification results to that of the state-of-the-art EM-based approach [Wang et al. 2013a], where simple heuristic rules are handcrafted in advance and each car makes local claims about the regulator type of each intersection it passes. All these claims are then aggregated at a central server where the EM algorithm is run to compute the most likely reliability of cars and the regulator types of intersections. Since we have multiple schemes, we pick, for ease of demonstration, the eFA (economic feature aggregation, and with 20% ground-truth training data) scheme for the comparison. As shown in Table II, our SmartRoad system not only outperforms the state-of-the-art approach across the board, but also displays much more stable performance among all regulator-typed intersections, as opposed to the EM-based approach, which, for example, achieves about 11% recognition false positive rates for traffic lights but almost 40% for uncontrolled intersections.

Please note that so far the experiments are only on the basic transmission and classification schemes, for which SmartRoad has already shown superior performance over the state-of-the-art approach. Later, more advanced classification schemes (with

Table II. Classification Performance Comparison

		Traffic Lights	Stop Signs	Uncontrolled
True positive rate	SmartRoad	80.6%	79.8%	85.3%
	EM-based	66.7%	76.8%	79.1%
False positive rate	SmartRoad	10.4%	4.9%	14.2%
	EM-based	10.7%	14.6%	36.9%

Single class classification results comparison between the state-of-the-art EM-based method and our SmartRoad system.

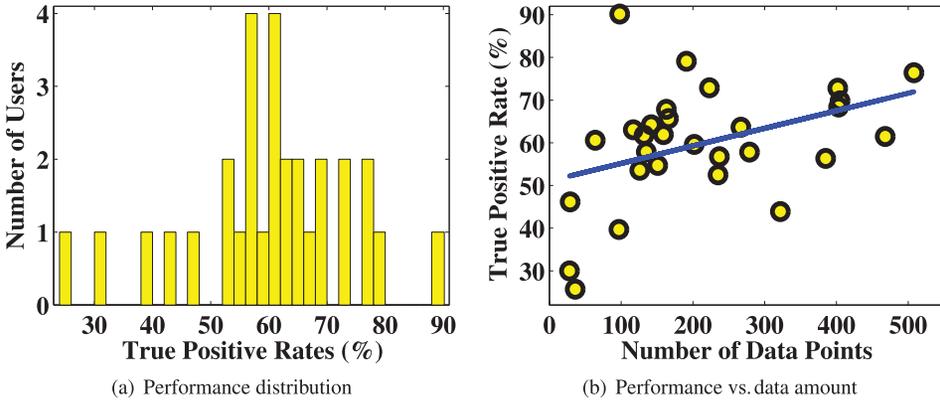


Fig. 9. Individual user local classifications.

active learning and self-training adapters enabled) are evaluated in Sections 5.1.5 and 5.1.6, which show even higher performance.

5.1.4. Individual User Classification Performance. We now turn our attention to individual users' classification performance. Figure 9(a) shows the distribution of classification accuracies for all the participating users, where only 10% data is used for training. As clearly shown, huge individual differences are observed among users; with two of them experiencing accuracies lower even than chance 33%, while one of them shows around 90% accuracy. This big variance and generally poor classification performance demonstrate how unreliable individual users can be. As also demonstrated in Figure 7, even with much higher training data availability, the average individual user performance still lags behind that of our various information aggregation schemes.

We also take one step further to investigate the correlation between users local classification performance and the amount of data they individually collect. The result is shown in Figure 9(b), from which it can be seen that even though the relationships scatter around quite a lot, a general trend of more-data-means-higher-performance is still observed, as illustrated by the linearly fit straight line.

5.1.5. Evaluating Active Learning Adapter. All previous results are from experiments where our active learning and self-training adapters are both deactivated. We next evaluate and discuss how the active learning and self-training adapters boost our classification engine's performance.

The experiment scenario is as follows. The system is deployed with the absolutely minimum training data availability, namely, three randomly selected labeled data, one for each class. As time elapses, data from participating users begin to show up and accumulate at the server. The server then periodically goes through training, classification, and adaptation phases, and dynamically adjusts itself based on the active learning and/or self-training adapter. The system continuously iterates like this, until

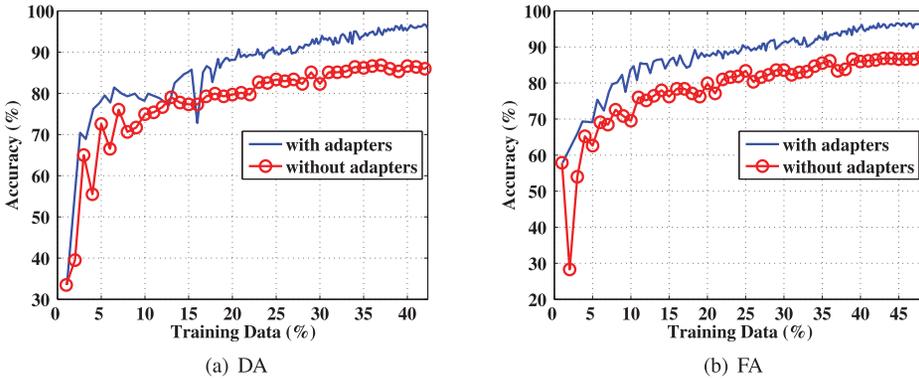


Fig. 10. Classification performance with active learning adapter.

all labels for all intersections are claimed by the server to have been classified with high confidence, which we consider the convergence of our system, or from a real-life participatory sensing application deployment point of view, the completion of the application tasks.

Having clearly specified the experiment scenario, we now look at the experiment results in detail. Figure 10 shows the classification performance for DA and FA schemes with the active learning adapter (see Algorithm 1) activated. The two schemes yield very similar results, thus we only focus on one of them. We look at the DA scheme result as shown in Figure 10(a). As seen, the x -axis represents the growth of the training dataset as a result of the active learning adapter taking effect and actively requesting for manual labels be acquired for the intersection data points with low classification confidence scores from the previous iteration. The blue trace indicates the evolution of the classification accuracy of the resultant classification model at all iterations, where the red one illustrates the baseline classification model trained from the same amount of training data, which are selected at random instead of by the active learning adapter. It is easily seen that, starting with only three randomly selected labeled data points that correspond to less than 1% of our already small dataset, our system quickly climbs up to 80% accuracy with only 5% of data used for training under the guidance of the active learning adapter. The advantage of our system over the baseline classification scheme is maintained throughout the entire experiment iteration duration. Upon application sensing task completion, our system claims a more than 10% advantage margin over the baseline, reaching over 95% accuracy. Note that with only 20% data for training, our system already achieves about 90% accuracy, which should be more than satisfiable for most crowd-sourced participatory sensing application requirements. Note also that this performance is not achieved by the baseline classification scheme even at its convergence point with much higher amount of training data.

5.1.6. Evaluating Self-Training Adapter. We then look at how the self-training adapter (see Algorithm 2) helps our detection and identification system. Figure 11 shows the experiment results that are under the same settings as Figure 10 with the only difference being the activation of both active learning and self-training adapters for the classification tasks. Let us again just focus on the DA performance, shown in Figure 11(a). Comparing it to the one just discussed, we can see that the same set of characteristics are still present in our system, be it the clear advantage margins over the baseline method or the fast reaching of outstanding performance with small amount of training data. However, the most important difference brought about by the self-training adapter is the fast convergence compared to the previous experiment. As clearly shown,

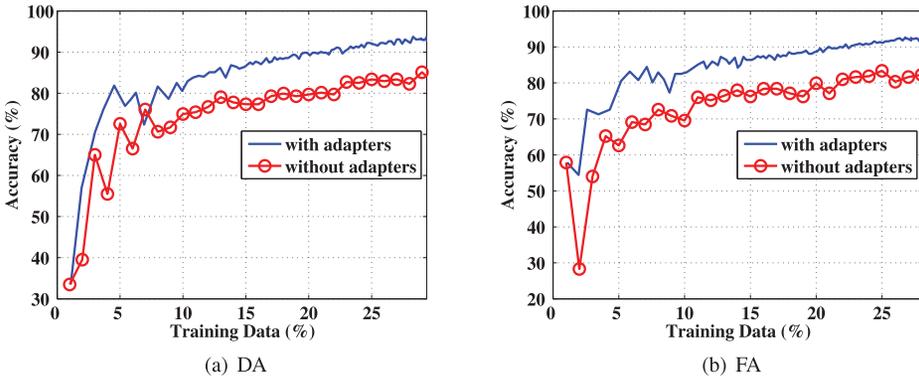


Fig. 11. Classification performance with both active learning and self-training adapters.

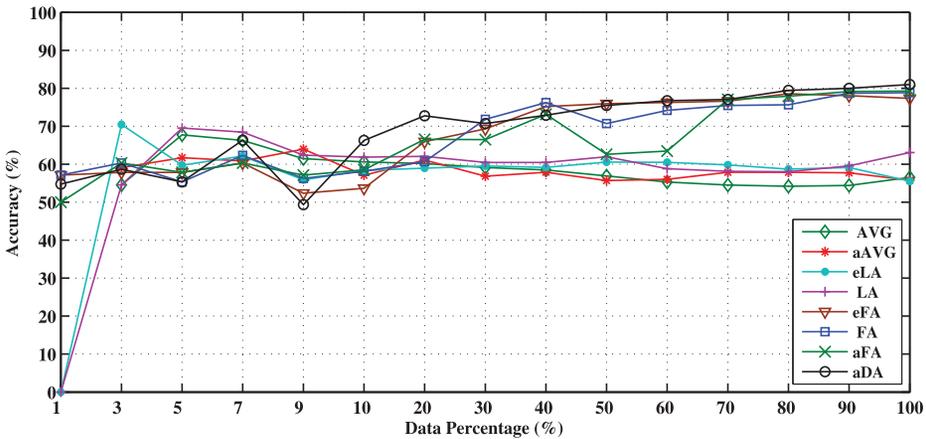


Fig. 12. The detection and identification performance of the unsupervised spectral clustering component.

when the active learning adapter is activated alone, the system requires a more than 40% training data ratio to converge, that is almost half of the entire data; however, when the self-training adapter is also activated, convergence is reached at about a 28% training data ratio, which is just slightly over one-quarter of the data, indicating a speedup of application completing its tasks by nearly 100%. The performance penalty is quite minimum, as can be seen from the figures. Please note that, due to the limited amount of our collected data, we would not consider the absolute “one-half” and “one-quarter” marks here as generally indicative for other participatory sensing applications and deployments; but the relative speedup and save on label requirement are surely promising.

5.2. Unsupervised Classification

5.2.1. Evaluating Unsupervised Classification Performance. The detection and identification performance of the unsupervised clustering component is shown in Figure 12. For the simplicity of presentation, we only show the clustering results on six different schemes, together with the measures of AVG and eAVG. As can be seen, the presented curves form two groups based on their performance levels, namely, aDA, aFA, FA, and eFA as one group and LA, eLA, AVG, and aAVG as the other. The curves in the first group illustrate accompanying classification performance that grows with the increase of

Table III. Cluster Feature Statistics

Clusters	A	B	C
Final stop duration (s)	11.28	2.50	0.12
Min crossing speed (m/s)	5.27	2.03	11.82

The mean statistics of the Final stop duration and Min crossing speed physical features for different clusters.

data amount, until reaching around 80% accuracy. This is quite valuable for a practical working system considering the zero availability of ground-truth information. On the other hand, the curves in the second group do not show much change during the accumulation of data. Obviously, the experimental results favor the schemes that place the clustering component at the central server.

Determining the final labels for the different clusters produced by the clustering algorithm is straightforward; taking a quick look at the average of the clustering result data, one can easily infer the corresponding label information by using common sense. For example, the clustering result of the aDA scheme produces the mean statistics as shown in Table III.

It is easily seen that clusters *A*, *B*, and *C* correspond to traffic light, stop sign, and uncontrolled intersections, respectively, as common sense tells us, for example, if a car does stop at a traffic regulator, then a red light would be the regulator that causes the longest wait; and when considering the lowest passing speeds a car demonstrates when crossing different intersections, at the uncontrolled ones we would observe the highest speed compared to traffic lights or stop signs.

6. DISCUSSION

Toward the goal of designing a participatory system that can easily be deployed at a large scale, we have focused on relying solely on GPS data in our system. We do, however, recognize and actually have considered augmenting our system with other orthogonal modalities, as taking advantage of multiple heterogeneous types will likely help improve detection and recognition accuracy. In this section, we briefly discuss several possibilities.

As also mentioned before, computer vision-based approaches have been widely studied. Though we have previously argued that vision-based approaches do not align with our design goals for various reasons, we do want to mention that, during navigation, some people do like to mount their phones in an upright position (as opposed to simply tossing their phones on the passenger seat or in the center console, etc.) such that the phone camera can have good view of the road and signs ahead, in which case, existing vision-based recognition methods can be borrowed here for opportunistic invocation to help augment the classification result from using GPS data alone.

Regarding motions of the car, which several of the features we extract from the GPS data were designed to capture, phone accelerometers can also be used to sense them; for example, the car acceleration and deceleration motions. In our preliminary experiment, we have also discovered that accelerometer data can be used to classify whether the car is idle or in motion as well. This information can certainly help with the traffic regulator detection task. However, this would only work if the phone is stationary relative to the car; otherwise (e.g., it is sliding on a slippery surface or is held in hand) the extraneous motion will likely mask the car's motion.

From the auditory perspective, using the microphone of the phone, we can train a sound-based classifier that recognizes the Accessible Pedestrian Signals (e.g., the sounds that are played for walk signals at traffic lights), the detection of which would indicate the existence of traffic lights, augmenting our current GPS-based method.

Being sound-based, however, this method would potentially be affected by the ambient noise on the road, people speaking in the car, etc.

Also, Geographic Information Systems (or GIS) can potentially be integrated into the back end to complement the crowd-sourced data of our system. For example, Open Street Map [Open Street Map 2014] data includes road type information (e.g., residential, highway, pedestrian, etc.). We have experimented with designing heuristics for predicting traffic regulator types based on the types of the roads intersecting, and have seen reasonable accuracies.

Above said, having the potential of improving detection accuracy, these various methods are not necessarily crucial components for our system, and shaping them such that our fundamental design goals are not violated is not a trivial task, which is likely worth substantial separate research efforts. We thus chose to focus on our self-contained SmartRoad system as discussed in this article, and explore extension possibilities, like the ones discussed previously, in our future work.

7. RELATED WORK

The increasing availability of sensors integrated in smartphones, such as GPS, accelerometer, gyroscope, and microphones, provides many new opportunities for mobile sensing applications. Compared with many traditional mobile sensing systems that rely on specialized sensing, computing, or communication devices, smartphones are more suitable for large deployment due to their popularity. Thus far, a large spectrum of smartphone-based sensing systems has been prototyped, covering various mobile sensing applications [Chandrasekaran et al. 2011; Eisenman et al. 2009; Hao et al. 2013; Hemminki et al. 2013; Miluzzo et al. 2010; Mun et al. 2009; Nath 2012; Nawaz et al. 2013; Qiu et al. 2011; Rana et al. 2010; Wang et al. 2013b, 2013c, 2013d; You et al. 2013]. Among these applications, road sensing is an important one and this article belongs to that category. Road sensing systems are normally dedicated to analyzing data collected by all kinds of sensing devices carried in vehicles. An early and representative road sensing test bed is CarTel [Hull et al. 2006], which is composed of a Soekris embedded computer running Linux, a WiFi card, a Sprint network card, an external GPS, and a three-axis accelerometer.

More recently, smartphones have become the major player in road sensing, and have been used in various scenarios. For example, researchers have shown, by using smartphone's internal sensors to monitor the in-vehicle dynamics, they can reliably determine the driver phone usage [Wang et al. 2013d]; ParkSense [Nawaz et al. 2013] uses smartphones and leverages the ubiquitous urban WiFi beacons to sense drivers' returning and leaving street parking spots; eNav [Hu et al. 2015] exploits phone's motion sensor to drastically reduce GPS samplings, and provides energy efficient navigation services; CarSafe [You et al. 2013] uses the dual cameras on modern smartphones to detect and alert drivers to dangerous driving conditions and behaviors; smartphone-based opportunistic data sharing and offloading has also been studied for vehicular sensing scenarios [Hu et al. 2014]. In-vehicle smartphones were used to analyze traffic patterns in order to provide better navigation services. For example, VTrack [Thiagarajan et al. 2009] and CTrack [Thiagarajan et al. 2011] are two related systems that process error-prone position streams to accurately estimate trajectories and delays. They argue that GPS has several limitations such as unavailability for some phones, deficiencies in "urban canyons" (tall buildings and tunnels), and power-hungry nature. Instead, they rely on an alternative, less energy-hungry but noisier sensory data like WiFi and cellular fingerprints. They both match a sequence of observations rather than a single one using constraints on the transitions a moving vehicle can make between locations. Their difference is that VTrack first converts fingerprints to coordinates, while CTrack matches fingerprints directly using a two-pass Hidden Markov Model (HMM).

With the popularization of GPS equipped smartphones, GPS sensors are becoming more preferred for road sensing applications such as vehicle tracking [Biagioni et al. 2011], due to their accuracy. Two closely related works to our SmartRoad are Carisi et al. [2011] and Wang et al. [2013a]. They also work with GPS traces in detecting traffic lights and stop signs on roads. Carisi et al. [2011] heavily uses map information, and focus on a binary classification problem on a small set of manually selected candidate locations that are already either traffic light or stop sign controlled. Wang et al. [2013a] also only looks at detecting traffic light and stop sign, and does so separately. Both works use rule-based methods that require either exhaustive testing on a complete labeled dataset or prior knowledge that might not be globally applicable. In contrast, our SmartRoad system considers all road intersection locations ever visited by any user, and detects both regulator controlled and uncontrolled intersections, based on statistical classification approach for automatic model generation without requiring a large amount of labeled data or any human-specified rules.

There exists other road sensing work, studying the problem of recognizing road traffic signs [de la Escalera et al. 2003; Koukoumidis et al. 2011; Maldonado-Bascon et al. 2007]. This work solves a similar problem to ours. Different from SmartRoad, however, these schemes all rely on windshield-mounted cameras to opportunistically capture road signs that the vehicle has passed by, and use vision techniques to detect and classify the signs from the video recorded by the cameras. For example, Signal-Guru [Koukoumidis et al. 2011] is a software service running on mobile smartphones. It is designed for predicting the traffic signal schedule through analyzing the video recorded by the phone camera. Another work [Maldonado-Bascon et al. 2007] proposes a Support Vector Machine (SVM) based road sign detection and recognition system that can detect and recognize signs with different color and shape. Compared with these systems, SmartRoad provides a much more lightweight system that does not require users to explicitly participate in sensing (e.g., by taking photos) and does not entail modifications to their vehicles (such as mounting a camera). Vision-based sensing suffers from two further limitations: (i) Video shooting and processing are more computationally expensive, and thus would result in more energy consumption. (ii) Due to the dynamics of road sign placement, in-vehicle cameras may not be able to capture all the signs passed by the vehicles, resulting in some miss detections.

8. CONCLUSION

In this article, we present the design, implementation, and evaluation of SmartRoad, a crowd-sourced traffic regulator detection and identification system that uses simple GPS trace data to determine the presence or absence of stop signs or traffic lights at all road intersections. We deploy the system in 35 volunteer users' vehicles for two months, collecting 4,000 miles of driving data covering hundreds of unique intersection locations. Experiment results show that, in addition to taking advantage of the participatory sensing paradigm, SmartRoad is able to adapt to various application requirements and evolve to fit under different sensing conditions. It carries out and completes the detection and identification tasks in a robust, effective, and efficient manner, achieving outstanding detection and identification performance on task completion.

REFERENCES

- Apple iOS. 2014. <http://www.apple.com/ios/>.
- Audi Travolution. 2010. <http://www.audiusanews.com/newsrelease.do?id=1812>.
- J. Biagioni, T. Gerlich, T. Merrifield, and J. Eriksson. 2011. Easytracker: Automatic transit tracking, mapping, and arrival time prediction using smartphones. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys'11)*. 68–81.

- L. Breiman. 2001. Random forests. *Machine Learning* 45, 1 (2001), 5–32.
- R. Carisi, E. Giordano, G. Pau, and M. Gerla. 2011. Enhancing in vehicle digital maps via GPS crowdsourcing. In *WONS*.
- Celery. 2014. <http://celeryproject.org/>.
- G. Chandrasekaran, T. Vu, A. Varshavsky, M. Gruteser, R. P. Martin, J. Yang, and Y. Chen. 2011. Tracking vehicular speed variations by warping mobile phone signal strengths. In *PerCom*. IEEE, 213–221.
- S. Dasgupta and J. Langford. 2009. A tutorial on active learning. *International Conference on Machine Learning*.
- A. de la Escalera, J. M. Armingol, and M. Mata. 2003. Traffic sign recognition and analysis for intelligent vehicles. *Image and Vision Computing* 21, 3 (2003), 247–258.
- Django. 2014. <https://www.djangoproject.com//>.
- S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G. S. Ahn, and A. T. Campbell. 2009. BikeNet: A mobile sensing system for cyclist experience mapping. *ACM Transactions on Sensor Networks (TOSN)* 6, 1 (2009), 6.
- Galaxy Nexus. 2013. Homepage. Retrieved from <http://www.android.com/devices/detail/galaxy-nexus/>.
- R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher. 2010. GreenGPS: A participatory sensing fuel-efficient maps application. In *MobiSys*.
- Garmin. 2013. Garmin: Choose Your Country. Retrieved from <http://www8.garmin.com/buzz/ecoroute/>.
- Google Android. 2013. Homepage. Retrieved from <http://www.android.com/>.
- Google Maps. 2014. Homepage. Retrieved from <https://www.google.com/maps/preview/>.
- Google. Google Self-Driving Car Project. 2015. <http://www.google.com/selfdrivingcar/>.
- J. Han, M. Kamber, and J. Pei. 2011. *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann.
- T. Hao, G. Xing, and G. Zhou. 2013. iSleep: Unobtrusive sleep quality monitoring using smartphones. In *SenSys*. ACM, 4.
- S. Hemminki, P. Nurmi, and S. Tarkoma. 2013. Accelerometer-based transportation mode detection on smartphones. In *SenSys*. ACM, 13.
- S. Hu, H. Liu, L. Su, H. Wang, T. F. Abdelzaher, P. Hui, W. Zheng, Z. Xie, and J. Stankovic. 2014. Towards automatic phone-to-phone communication for vehicular networking applications. In *INFOCOM*.
- S. Hu, L. Su, S. Li, S. Wang, C. Pan, S. Gu, T. Amin, H. Liu, S. Nath, R. R. Choudhury, and T. F. Abdelzaher. 2015. Experiences with eNav: A Low-power vehicular navigation system. In *UbiComp*.
- B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. 2006. CarTel: A distributed mobile sensor computing system. In *SenSys*.
- E. Koukoumidis, L. S. Peh, and M. R. Martonosi. 2011. SignalGuru: Leveraging mobile phones for collaborative traffic signal schedule advisory. In *MobiSys*.
- A. Krause and C. Guestrin. 2007. Nonmyopic active learning of Gaussian processes: An exploration-exploitation approach. In *ICML*. ACM, 449–456.
- S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, and F. López-Ferreras. 2007. Road-sign detection and recognition based on support vector machines. *IEEE Transactions on Intelligent Transportation Systems* 8, 2 (2007), 264–278.
- E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell. 2010. Darwin phones: The evolution of sensing and inference on mobile phones. In *MobiSys*.
- T. M. Mitchell. 1997. *Machine Learning*. McGraw-Hill.
- M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda. 2009. PEIR, the personal environmental impact report, as a platform for participatory sensing systems research. In *MobiSys*.
- S. Nath. 2012. ACE: Exploiting correlation for energy-efficient and continuous context sensing. In *MobiSys*. ACM, 29–42.
- S. Nawaz, C. Efstratiou, and C. Mascolo. 2013. ParkSense: A smartphone based sensing system for on-street parking. In *MobiCom*. ACM, 75–86.
- A. Y. Ng, M. I. Jordan, Y. Weiss, and others. 2002. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems* 2, 849–856.
- Open Street Map. 2014. <http://www.openstreetmap.org/>.
- J. Qiu, D. Chu, X. Meng, and T. Moscibroda. 2011. On the feasibility of real-time phone-to-phone 3D localization. In *SenSys*. ACM, 190–203.
- R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu. 2010. Ear-phone: An end-to-end participatory urban noise mapping system. In *IPSN*. ACM, 105–116.

- F. Saremi, O. Fatemeh, H. Ahmadi, H. Wang, T. Abdelzaher, R. Ganti, H. Liu, S. Hu, S. Li, and L. Su. 2015. Experiences with greengps—fuel-efficient navigation using participatory sensing. *IEEE Transactions on Mobile Computing (TMC)*.
- A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. 2009. VTrack: Accurate, energy-aware road traffic delay estimation using mobile phones. In *SenSys*.
- A. Thiagarajan, L. S. Ravindranath, H. Balakrishnan, S. Madden, and L. Girod. 2011. Accurate, low-energy trajectory mapping for mobile devices. In *NSDI*.
- D. Wang, T. Abdelzaher, L. Kaplan, R. Ganti, S. Hu, and H. Liu. 2013a. Exploitation of physical constraints for reliable social sensing. In *RTSS*. IEEE.
- H. Wang, X. Bao, R. R. Choudhury, and S. Nelakuditi. 2013b. InSight: Recognizing humans without face recognition. In *HotMobile*. ACM, 7.
- Y. Wang, X. Liu, H. Wei, G. Forman, C. Chen, and Y. Zhu. 2013c. CrowdAtlas: Self-updating maps for cloud and personal use. In *MobiSys*. ACM.
- Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, and R. P. Martin. 2013d. Sensing vehicle dynamics for determining driver phone use. In *Mobisys*. ACM.
- Waze. 2014. Waze Social GPS Maps Traffic. Retrieved from <https://play.google.com/store/apps/details?id=com.waze>.
- C.-W. You, N. D. Lane, F. Chen, R. Wang, Z. Chen, T. J. Bao, M. Montes-de Oca, Y. Cheng, M. Lin, L. Torresani, and A. Campbell. 2013. CarSafe App: Alerting drowsy and distracted drivers using dual cameras on smartphones. In *MobiSys*. ACM.
- X. Zhu. 2008. Semi-Supervised Learning Literature Survey. Computer Science Technical Report. University of Wisconsin-Madison.

Received December 2013; revised October 2014; accepted April 2015