SmartRoad: A Crowd-Sourced Traffic Regulator Detection and Identification System

Shaohan Hu, Lu Su, Hengchang Liu, Hongyan Wang, Tarek Abdelzaher
Department of Computer Science, University of Illinois at Urbana-Champaign
Email: {shu17, lusu2, hl4d, wang44, zaher}@illinois.edu

ABSTRACT

In this paper we present SmartRoad, a crowd-sourced 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, SmartRoad addresses various challenges in participatory sensing scenarios, including data unreliability/sparsity, energy constraints, and the general lack of ground truth information. SmartRoad automatically adapts to different application requirements by intelligently choosing the most appropriate information representation and transmission schemes; it also dynamically evolves its core detection and identification engines to effectively take advantage of any external ground truth information or opportunity. With these two characteristics, SmartRoad consistently delivers outstanding performance for its road sensing tasks. We implement SmartRoad on a vehicular smartphone testbed, and deploy on 35 external volunteer users' vehicles for two months. Experiment results show that SmartRoad can robustly, effectively and efficiently carry out its detection and identification tasks without consuming excessive communication energy/bandwidth or requiring too much ground truth information.

1. INTRODUCTION

Traffic regulators, such as stop signs and traffic lights, are designed to regulate competing flows of traffic at intersections. They are among the most commonly used traffic control signals, and play significant roles in people's daily driving behaviors. Despite the safety and convenience benefits they bring, the stop signs and traffic lights do charge their toll. The stop-and-go movement pattern of vehicles caused have resulted in substantial increase of gas consumption and CO_2 emissions [2]. Driven by this problem, some recent efforts are taken to reduce the negative effects, such as EcoRoute [13] and GreenGPS [12]. Navigation services like this need to take into account the actual locations of traffic lights and stop signs. Yet, unlike the case with road-maps, no nationwide database exists today that documents traffic light and stop sign locations. Instead, this information is quite fragmented, buried in physical archives of different counties and municipalities.

To address the above challenge, in this paper, 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 to 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 street-view.

While 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), it was quickly discovered that some locations (such as the authors' own neighborhood) do not have adequate street-view coverage, rendering the approach less valuable. Instead, we focus on achieving accurate detection and identification performance using mobile phones. Towards that end, SmartRoad addresses the following major challenges.

Unreliable Data: The raw data collected by phone sensors in individual vehicles are often unreliable. Many factors potentially affect the quality of sensory data. Examples are the quality of the sensing devices, the noise level of the physical environment, and sometimes the ways in which the sensing devices are used. To address this challenge, SmartRoad combines information acquired from multiple vehicles to reach a more accurate result. Here the underlying philosophy is: 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.

Sparse Data: Data sparsity imposes another challenge to the task of detecting and identifying traffic regulators. Though the total number of collected traces might be large, the amount of information corresponding to individual intersec-

tions is limited. To tackle this problem, SmartRoad launches another dimension of information aggregation, i.e., it combines the data traces at all the intersections to investigate the common moving patterns of vehicles in the presence or absence of stop signs or traffic lights.

Expensive Data: Since users pay real money for communication energy and 3G 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 places raw data processing operations, which are all lightweight, on client phones. Information that are then sent to the server takes much smaller size compared to raw data.

Unlabeled Data: In the area of data mining and machine learning [23, 16], labeling is a common concept widely used in the task of classification. In this paper, a label corresponds to the ground truth information regarding the presence or absence of stop sign or traffic light at a particular intersection. Since our sensors work in a passive mode, they do not provide any label information, nor do we assume to be able to receive any label information from participating users. We do, however, design and execute our SmartRoad system in a way that it can intelligently take advantage of any level of label information. Furthermore, whenever budget allows, SmartRoad is able to actively evolve its core detection and identification engines to substantially improve performance.

The general design of SmartRoad follows a client-server framework. We place the client component on a vehicular smartphone testbed, and the server component on a workstation. Recent years have witnessed the popularity explosion of smartphones. Several features of smartphones make them an appealing platform for our SmartRoad system. i) Smartphones are packed with an array of different sensors on board, such as GPS, accelerometers, and barometers, ii) Smartphone operating systems [18, 1] allow users to install and run third-party applications that can potentially take advantage of the on board sensors and provide services to the users, such as GPS-assisted local search and navigation, and 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 above features make smartphones an ideal platform for our SmartRoad system.

We implement SmartRoad on a vehicular smartphone testbed, and deploy it on 35 external volunteer users' vehicles. Through an experiment of two months collecting around 4000 miles of driving data containing hundreds of regulator-controlled and uncontrolled locations, we demonstrate that SmartRoad can deliver outstanding detection and identification performance without consuming excessive communication energy/bandwidth or requiring too much ground truth information.

2. SYSTEM OVERVIEW

In this section, we provide an overview of our SmartRoad participatory sensing system that carries out the traffic regulator detection and identification tasks. SmartRoad contains three modules: a data acquisition module, a detection and identification module, and a feedback module. They are deployed on two different platforms distributed in-vehicle deployed smartphones, and a central server. Figure 1 illustrates the architecture overview of the SmartRoad system. We next discuss each of these three modules in more detail.

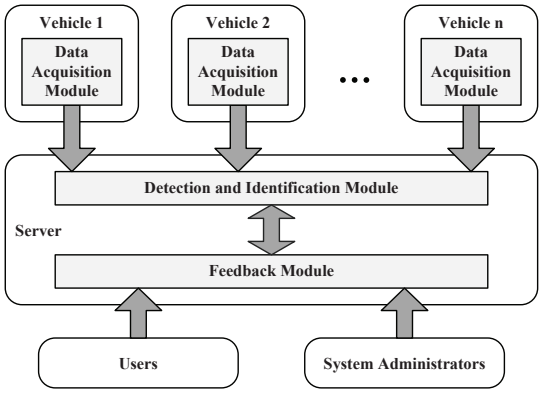


Figure 1: Architecture of the SmartRoad system

2.1 Data Acquisition Module

As shown in Fig. 1, the data acquisition module is installed in smartphones deployed in vehicles. It performs four major tasks as listed below.

Data Collection: Access the various on board sensors to sense the physical environment.

Data Processing: Process 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 communication energy and bandwidth.

Data Delivery: Upload the processed data to the central server. Cellular data networks and open public WiFi networks are both valid options as the transmission channels. However, they both have certain limitations such as being potentially monetarily costly or not generally available. SmartRoad gives users the choices to use either or both communication channels based on their own personal situations and preferences.

Energy Management: Besides the tasks mentioned above, the data acquisition module also needs to monitor and manage its energy usage in order not to affect the other phone functionalities or the general user experience.

2.2 Detection and Identification Module

As shown in Fig. 1, the detection and identification module resides on the central server, and processes data received

from individual vehicles. Thus its role is twofold:

Data Server: Listen for incoming connections and receive data from in-vehicle phones. Data can be either directly piped to the final processing engine, or stored in database for later batch processing.

Central Processor: Carry out detection and identification tasks, i.e., make 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.

2.3 Feedback Module

The feedback module also resides on the central server. As illustrated in Fig. 1, the information flow between the detection module and the feedback module is bidirectional, corresponding to its two major tasks.

Outputs to User: Take as input the detection results and present them to next tier applications that take as input the detection and identification results, for example, assisted driving or navigation systems such as GreenGPS [12]. We also visually present detection and identification results via a web service interface.

Feedback from User: The web service interface can also be used by system administrators and users to correct detection errors or provide ground truth information, which is then sent back to the detection and identification module for dynamic and adaptive performance improvement, as discussed in detail in Section 3.

3. DETECTION AND IDENTIFICATION

To tackle the traffic regulator detection and identification problem, the most intuitive approach would be to use rule-based methods, where we specify a set of rules regarding vehicles' moving patterns, and then use these rules to derive detection and identification decisions from the driving data. For example, we can come up with rules like the following: i) Intersections where more than 90% cars come to full stop before crossing have stop signs, or ii) Intersections where users' average crossing speed is less than 5 mph have stop signs. Both rules sound quite reasonable and intuitive; They, however, lead us to think about the following two questions that we need to answer if we want to be able to effectively and systematically build such rules:

- *What decision points should be selected for the rules?* In our example above, we use 90% and 5 mph as the decision points, but why not 80% and 3 mph?
- *What set of rules should be used?* Between Rule i and rule ii in our example above, which one is better? Also, can we somehow use both of them?

A naïve solution to these questions would be to exhaustively check all combinations of all the rules and all the possible values for the decision points. This would, however, result in enormous computational complexity, rendering it infeasible in practice. Moreover, it is also possible that a set of rules is only applicable for a certain sub-region of the problem space but not others. For example, it is possible that the rules in our example above work quite well for a rural area, but fail miserably under an urban setting. Therefore, an effective and robust detection and identification system needs to have the capability of automatically selecting appropriate rules and the corresponding decision points.

Towards this goal, we adopt statistical classification techniques [23] as the core detection and identification components for our SmartRoad system, which, given data collected from the target problem domain, automatically explore the problem space and select the most suitable set of rules and decision points for the detection and identification tasks.

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

3.1 Data Preprocessing

Trace Extraction: The raw GPS trace data are broken into separate traces by dividing between consecutive GPS frames where the timestamp difference exceeds a threshold (corresponding to users parking or powering off cars for the night). Each trace then represents a separate continuous driving activity. Whenever the client phone-side data acquisition module detects the end of a trace, all preprocessing and subsequent feature extraction (as will be discussed shortly) operations are carried out for that trace.

Trace Segmentation: Each trace is scanned through and broken into segments by using 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 [15] data, and it is the only map information that we use in our entire system.

Turning Removal: Trace segments involving cars making turns can potentially confuse a detection and identification system, since the resulting approach→stop→turn movement pattern 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 turnings. 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 degrees, between the two. Straight crossing

segments are therefore preserved.

Intersection Decomposition: Intersections generally have three or more incident approaching directions, which might or might not have different traffic regulator situations. For example, a 4-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 4-way intersection in our previous example would be decomposed to 4 different ones, the traffic regulator situations of which are to be considered separately.

3.2 Feature Extraction

Given the preprocessed data, the next step is to derive 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 extracting features from the GPS data. We extract 5 features, all of which are inspired by everyday common sense (some are also partially used by prior work [5]). Next we give detailed descriptions of the intuition and extraction for all 5 features.

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 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 duration 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.

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.

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.

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 make multiple rounds of stop-and-go actions. To extract, we simply count the number of continuous 0-speed blocks in the segment.

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 last 0-speed block, if any, in the segment and compute its distance from the intersection point.

Note that all preprocessings and feature extractions are done locally on phones. As they are all extremely lightweight operations, the computation energy consumption is negligible.

Our system extracts the 5 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 5 physical dimensions. In this paper, 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 components, namely *supervised classification* and *unsupervised classification*, that carry out the actual detection and identification tasks under various settings. As suggested by the names, their applicabilities depend on the availability of training data set, 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 paper, 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. In the next two subsections, we provide detailed explanation and discussion of these two classification components.

3.3 Supervised Classification

In the data set 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 [4] 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 are much like the derivation of rules, but in an automatic and systematic fashion. Provided with labeled training data, each decision tree iteratively figures out which feature and what decision boundary value to use according to the labeled training data.

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 below.

Active Learning Adapter: In realistic participatory sensing application scenarios, it is feasible that budget allows to manually acquire ground truth information, but only up to some small amount 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 get their ground truth information in order to maximize our final 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. This scheme is easy to carry out, but does not necessarily result in satisfactory performance. 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 these particular intersections. Here we borrow the active learning philosophy [8, 20]. 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 7 votes for traffic lights and 3 for uncontrolled, then the final classification result label will be traffic lights, with a confidence score of 0.7.

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 [31], looks at its own past classification results and try 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 is that classification results with high classification confidences are most likely to be correct, and thus including these data points into the training set will likely help expedite the overall classification tasks. Algorithm 2 demonstrates how our self training adapter works.

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 data set $\mathcal{D}_{\text{train}}$; 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

Initialize empty set $\mathcal{L}_{\text{conf}}$
while $|\mathcal{D}_{\text{target}}| > 0$ **do**
  train $\mathcal{C}_{\text{classify}}$ on $\mathcal{D}_{\text{train}}$
  apply $\mathcal{C}_{\text{classify}}$ on $\mathcal{D}_{\text{target}}$ to get result labels $\mathcal{L}$ and the corresponding confidence scores $\mathcal{S}$
  for $i \leftarrow 0$ to $|\mathcal{D}_{\text{target}}|$ **do**
    if $\mathcal{S}[i] \geq h_{\text{conf}}$ **then**
      $\mathcal{L}_{\text{conf}} \leftarrow \mathcal{L}_{\text{conf}} \cup \{\mathcal{L}[i]\}$
      $\mathcal{D}_{\text{target}} \leftarrow \mathcal{D}_{\text{target}} \setminus \{\mathcal{D}_{\text{target}}[i]\}$
    else if $\mathcal{S}[i] \leq l_{\text{conf}}$ **then**
      manually label $\mathcal{D}_{\text{target}}[i]$
      $\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_{\text{train}} \cup \{\mathcal{D}_{\text{target}}[i]\}$
      $\mathcal{D}_{\text{target}} \leftarrow \mathcal{D}_{\text{target}} \setminus \{\mathcal{D}_{\text{target}}[i]\}$
return $\mathcal{L}_{\text{conf}}$

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 data set $\mathcal{D}_{\text{train}}$; high confidence threshold h_{conf}
Output: High confidence label set $\mathcal{L}_{\text{conf}}$ that holds all final classification results

Initialize empty data set $\mathcal{L}_{\text{conf}}$
while $|\mathcal{D}_{\text{target}}| > 0$ **do**
  train $\mathcal{C}_{\text{classify}}$ on $\mathcal{D}_{\text{train}} \cup \mathcal{L}_{\text{conf}}$
  apply $\mathcal{C}_{\text{classify}}$ on $\mathcal{D}_{\text{target}}$ to get result labels $\mathcal{L}$ and the corresponding confidence scores $\mathcal{S}$
  for $i \leftarrow 1$ to $|\mathcal{D}_{\text{target}}|$ **do**
    if $\mathcal{L}_{\text{score}}[i] \geq h_{\text{conf}}$ **then**
      $\mathcal{L}_{\text{conf}} \leftarrow \mathcal{L}_{\text{conf}} \cup \{\mathcal{L}[i]\}$;
      $\mathcal{D}_{\text{target}} \leftarrow \mathcal{D}_{\text{target}} \setminus \{\mathcal{D}_{\text{target}}[i]\}$;
return $\mathcal{L}_{\text{conf}}$

3.4 Unsupervised Classification

Towards designing a robust system that functions even under extreme conditions where no ground truth information is available, we also develop 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 select spectral cluster-

ing [26] as the base unsupervised clustering algorithm. As opposed to traditional clustering algorithms like k-means or generative EM framework that always result in convex sets, spectral clustering can solve problems in much more complex scenarios because 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 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 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 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 5 different physical features, and derive statistical features from each of them. The resultant statistical features associated with each intersection are used as the input for 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.

Data Aggregation (DA): The original physical feature vector extracted for every crossing through an intersection is transmitted to the server. The server collects all physical feature vectors corresponding to the same intersection and computes a statistical feature vector for that intersection.

Feature Aggregation (FA): The per intersection statistical features are computed by each individual users, and are then opportunistically sent to the server, along with the associated crossing segment count information. All these information from different users are 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.

Label Aggregation (LA): Each individual user carries out classification locally on phone and only sends the classification result labels to the server, who then derives the final labels by carrying out where 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 3 sets of statistical feature selection schemes. As discussed below.

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.

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.

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 3 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 2.

Figure 2: The data acquisition module running on a Galaxy Nexus smart phone, deployed in car.

4.1 Data Acquisition Module

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 [14]. 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 to 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 form a DTN network 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

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 implemented in Django [10]. It provides a graphical user interface for easy interactions. It uses the python plugin Celery [6] to periodically update the traffic regulator detection and identification results as well as 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 EVALUATIONS

We deploy SmartRoad system for 35 external volunteer users. The in-vehicle client phones work in completely autonomous manners. All users are asked to drive just as how they normally do. The entire experiments last for about two months with individual users averaging about 3 weeks each. Roughly 4000 miles of driving data are collected in total, covering 158 traffic lights, 77 stop signs, and 228 uncontrolled unique intersections without considering turning segments. For each of the 3 regulator situations, The 25th, 50th, and 75th percentiles for the number of crossing segments are $\{6, 14, 34\}$, $\{1, 4, 11\}$, and $\{2, 10, 26\}$ for traffic lights, stops signs, and uncontrolled intersections, respectively. In order to have accurate ground truth information for evaluation purposes, we make several attempts to acquire such information from our municipalities but receive no useful response; Thus a *great* amount of time and effort are spent on manually physically collecting and verifying the ground truth for all the intersections.

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% true positive rate (TP) can be achieved without any ground truth information; 90% TP is reached with only 20% ground truth data at about 8% false positive rate (FP). Under realistic sensing application scenarios, SmartRoad claims and maintains a 5% to 10% advantage margin for both TP and FP measures over the baseline approaches at all levels of ground truth availabilities. SmartRoad completes the detection and identification tasks, confidently claiming labels for all intersections, with no more than 28% ground truth information, achieving outstanding classification performance of around 93% TP and 5% FP rates, which the baseline approach barely reaches with even twice the amount of ground truth information.

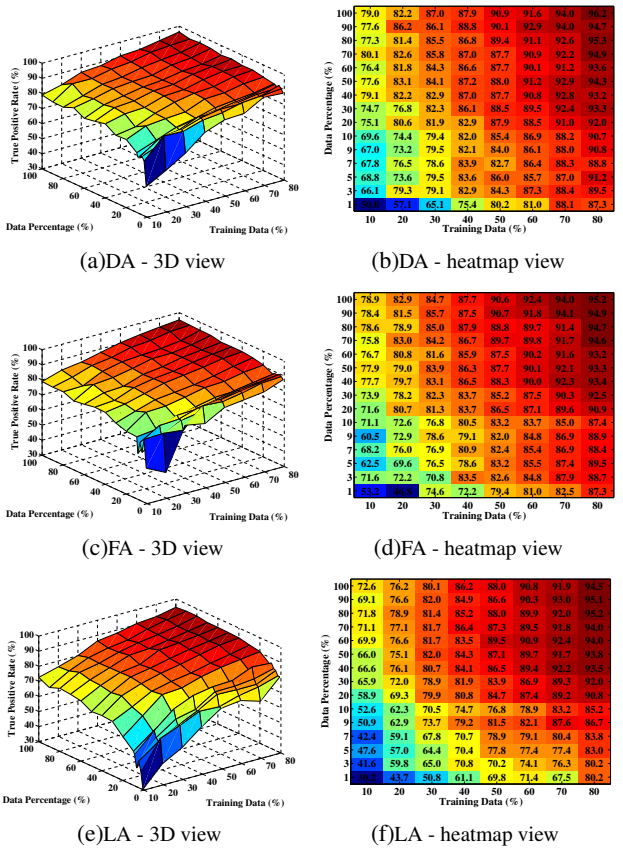
Detailed experiment results are presented as follows.

5.1 Supervised Classification

Evaluating Information Aggregation Schemes: First, we evaluate the classification performance for the 3 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 3 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 Fig. 3. It can be observed from these figures that for all data amounts or training data availability



(a)DA - 3D view (b)DA - heatmap view

(c)FA - 3D view (d)FA - heatmap view

(e)LA - 3D view (f)LA - heatmap view

Figure 3: Supervised classification performance

situations, 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 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.

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 5 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 Fig. 3 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 data set at its full size and examine the classification performance at varying

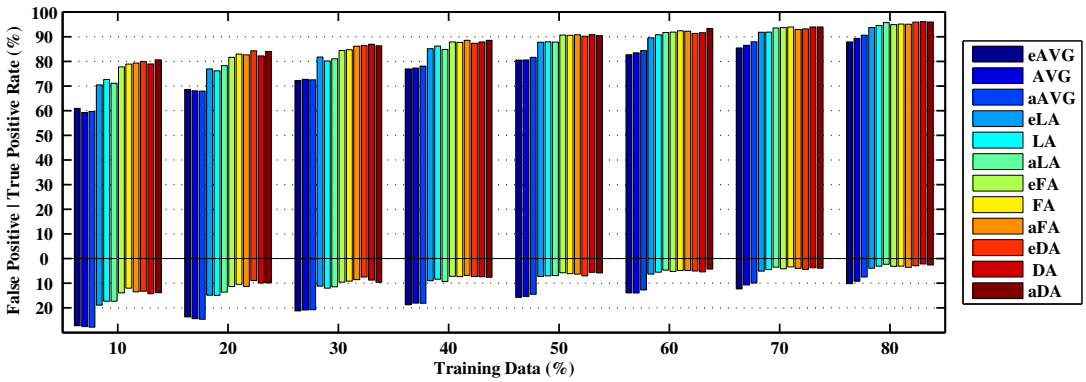


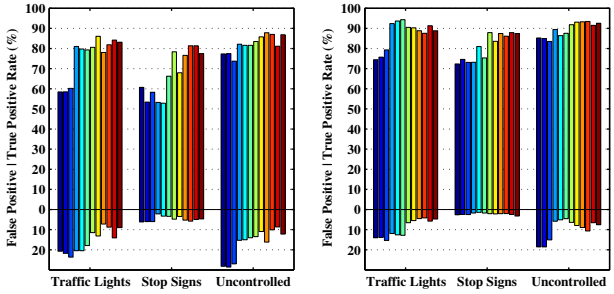
Figure 4: Classification performance at varying training data availabilities. The upward bar heights indicate the true positive rates while the downward false positive

training data availabilities and feature selection/information aggregation schemes, as Fig. 4 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 are 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, don't 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 3 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 true positive marks averaged at around 60% and false positives almost 30%. However, simple voting among the users would immediately improve the true positive rate to over 70% and reduce false positive rate to less than 20%; and any slightly more sophisticated scheme would further bump the true positive rate to around 80% and cut false positive rate down to 15% or less. This is a clear demonstration of the power of crowd-sourced participatory sensing, under considerable label constraints.

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 Fig. 5. We have the following two observations. i) The average classification performance for both training data availability levels agrees to the results shown in Fig. 4, and ii) Even though certain per class performance differences can be observed, it still is evident that SmartRoad classification engines treat all 3 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.



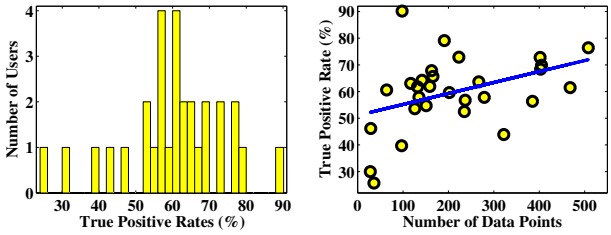
(a)Training on 20% ground truth (b)Training on 50% ground truth

Figure 5: The per class classification performance (using the same legend as Fig. 4)

Individual User Classification Performance: We now turn our attention to individual users' classification performance. Fig. 6(a) shows the distribution of classification true positive rates for all the participating users, where only 10% data is used for training. As clearly shown, huge individual differences are observed among users; with 2 of them experiencing accuracies lower even than chance 33%, while 1 of them showing around 90% accuracy. This big variance and generally poor classification performance demonstrate how unreliable individual users can be. As also demonstrated in Fig. 4, 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 Fig. 6(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 il-

lustrated by the linearly fit straight line.



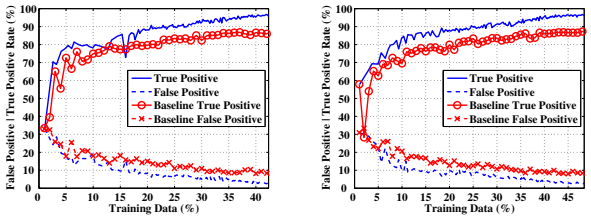
(a)Performance distribution (b)Performance v.s. data amount

Figure 6: Individual user local classifications

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 3 randomly selected labeled data, 1 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 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. Fig. 7 shows the classification performance for DA and FA schemes with the active learning adapter (see Alg. 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 the Fig. 7(a). As seen, the x-axis represents the growth of the training data set 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 two blue traces show the evolution of the true positive and false positive rates of the resultant classification model at all iterations, where the two red traces correspond to 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 3 randomly selected labeled data points which correspond to less than 1% of our already small data set, our system quickly reaches true positive rate of 80% with only 5% of data used for training under the guidance of 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% true positive advantage margin than the baseline, reaching 96%; and a 6% false positive advantage margin, reaching 3%. Note that with only 20% data for training, our system already achieves about 90% true positive rate and below 10% false positive rate, 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.



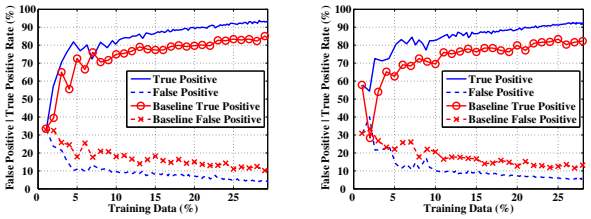
(a)DA (b)FA

Figure 7: Classification performance with active learning adapter

Evaluating Self Training Adapter: We then look at how the self training adapter (see Alg. 2) helps our detection and identification system. Fig. 8 shows the experiment results that are under the same settings as Fig. 7 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 Fig. 8(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, 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 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 speed up of application completing its tasks by nearly 100%. The performance penalties on both true positive and false positive rates are 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 speed up and save on label requirement are surely promising.

5.2 Unsupervised Classification

Evaluating Unsupervised Classification Performance: The detection and identification performance of the unsupervised clustering component is shown in Fig. 9. For the



(a)DA (b)FA

Figure 8: Classification performance with both active learning and self training adapters

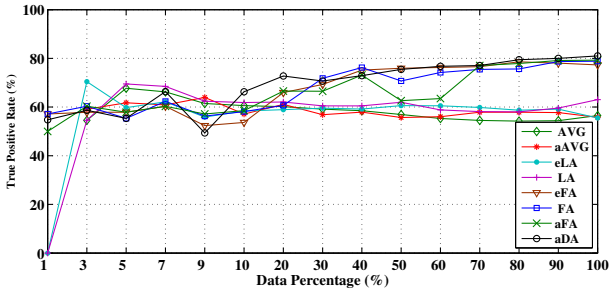


Figure 9: The detection and identification performance of the unsupervised spectral clustering component

simplicity of presentation, we only show the clustering results on 6 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 data amount, until reaching a true positive rate of around 80%. On the other hand, the curves in the second group do not show much changes 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 1.

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

Table 1: Mean statistics of the FinalStopDuration and MinCrossingSpeed physical features for different clusters

It is easily seen that Cluster 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 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. RELATED WORK

The increasing availability of sensors integrated in smart phones, such as GPS, accelerometer, gyroscope, and microphones, provides many new opportunities for mobile sensing applications. Compared with many traditional mobile sensing systems which rely on specialized sensing, computing, or communicating devices, smartphones are more suitable for large deployment due their popularity. Thus far, a large spectrum of smartphone-based sensing systems has been prototyped, covering various mobile sensing applications [24, 22, 28, 7, 27, 25]. Among these applications, road sensing is an important one and this paper 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 testbed is CarTel [17], which is composed of a Soekris embedded computer running Linux, a WiFi card, a Sprint network card, an external GPS, and a 3-axis accelerometer.

More recently, smartphones become the major player in road sensing, and have been used in various scenarios. For instance, BikeNet [11] provides a cyclist with data archival, retrieval, and visualization services through exploring interbicycle networking. In-vehicle smartphones were used to analyze traffic patterns in order to provide better navigation services. For example, VTrack [29] and CTrack [30] 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 [3], due to their accuracy. One closely related work to our SmartRoad is Carisi et al. [5]. They also work with GPS driving data, but heavily use map information. In contrast, SmartRoad only uses map intersection location information. Also they focus on a binary classification problem, namely traffic light v.s. stop signs, on a small set of manually selected candidate locations that are already either traffic light or stop sign controlled, whereas our SmartRoad system considers all intersections ever visited by all users, detects and identifies the traffic regulator type. In addition, for their data analysis, they use a rule based approach where rule decision points are selected by exhaustive testing on the complete data set with ground truth information, whereas SmartRoad uses statistical clas-

sification approach for automatic model generation, yielding satisfiable performance by using none or only a tiny amount of labeled data.

There exists other road sensing work, studying the problem of recognizing road traffic signs [19, 21, 9]. 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, SignalGuru [19] 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 [21] proposes an SVM (Support Vector Machine) based road-sign detection and recognition system which 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.

7. CONCLUSION

In this paper, 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 4000 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.

8. REFERENCES

[1] Android. http://www.android.com/.
[2] AudiTravolutionProject. http://www.audi.com/.
[3] J. Biagioni, T. Gerlich, T. Merrifield, and J. Eriksson. Easytracker: Automatic transit tracking, mapping, and arrival time prediction using smartphones. In *SenSys*, 2011.
[4] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
[5] R. Carisi, E. Giordano, G. Pau, and M. Gerla. Enhancing in vehicle digital maps via gps crowdsourcing. In *WONS*, 2011.
[6] Celery. http://celeryproject.org/.
[7] G. Chandrasekaran, T. Vu, A. Varshavsky, M. Gruteser, R. Martin, J. Yang, and Y. Chen. Tracking vehicular speed variations by warping mobile phone signal strengths. In *PerCom*, pages 213–221. IEEE, 2011.
[8] S. Dasgupta and J. Langford. A tutorial on active learning.
[9] A. de la Escalera, J. Armingol, and M. Mata. Traffic sign recognition and analysis for intelligent vehicles. *Image and vision computing*, 21(3):247–258, 2003.
[10] Django. https://www.djangoproject.com//.
[11] S. Eisenman, E. Miluzzo, N. Lane, R. Peterson, G. Ahn, and A. Campbell. Bikenet: A mobile sensing system for cyclist experience mapping. *ACM Transactions on Sensor Networks (TOSN)*, 6(1):6, 2009.
[12] R. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. Abdelzaher. GreenGPS: A participatory sensing fuel-efficient maps application. In *MobiSys*, 2010.
[13] Garmin. http://www8.garmin.com/buzz/ecoroute/.
[14] Google, Samsung. http://www.google.com/nexus/.
[15] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.
[16] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, third edition, 2011.
[17] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. Cartel: a distributed mobile sensor computing system. In *SenSys*, 2006.
[18] iOS. http://www.apple.com/ios/.
[19] E. Koukoumidis, L. Peh, and M. Martonosi. Signalguru: leveraging mobile phones for collaborative traffic signal schedule advisory. In *MobiSys*, 2011.
[20] A. Krause and C. Guestrin. Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. In *ICML*, pages 449–456. ACM, 2007.
[21] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, and F. López-Ferreras. Road-sign detection and recognition based on support vector machines. *Intelligent Transportation Systems, IEEE Transactions on*, 8(2):264–278, 2007.
[22] E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell. Darwin phones: the evolution of sensing and inference on mobile phones. In *MobiSys*, 2010.
[23] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
[24] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda. Peir, the personal environmental impact report, as a platform for participatory sensing systems research. In *MobiSys*, 2009.
[25] S. Nath. Ace: exploiting correlation for energy-efficient and continuous context sensing. In *MobiSys*. ACM, 2012.
[26] A. Ng, M. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.
[27] J. Qiu, D. Chu, X. Meng, and T. Moscibroda. On the feasibility of real-time phone-to-phone 3d localization. In *SenSys*, pages 190–203. ACM, 2011.
[28] R. Rana, C. Chou, S. Kanhere, N. Bulusu, and W. Hu. Ear-phone: an end-to-end participatory urban noise mapping system. In *IPSN*, pages 105–116. ACM, 2010.
[29] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *SenSys*, 2009.
[30] A. Thiagarajan, L. S. Ravindranath, H. Balakrishnan, S. Madden, and L. Girod. Accurate, Low-Energy Trajectory Mapping for Mobile Devices. In *NSDI*, 2011.
[31] X. Zhu. Semi-supervised learning literature survey, 2006.