ENHANCING VEHICLE DATA AVAILABILITY AND PRIVACY FOR CONNECTED CARS

BY

GORKEM KAR

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Marco Gruteser

And approved by

New Brunswick, New Jersey

May, 2017

ABSTRACT OF THE DISSERTATION

Enhancing Vehicle Data Availability and Privacy for Connected Cars

By GORKEM KAR

Dissertation Director:
Marco Gruteser

With the evolution of technology, vehicles are becoming increasingly connected and automated. They have evolved into rich sensing platforms with a plethora of diverse sensors, generating large amounts of real-time data including the location information, speed, acceleration, steering angle etc. With the recent advances of intelligent vehicle systems and Dedicated Short Range Communication (DSRC) devices, this data is broadcasted multiple times per second, so that each vehicle can be aware of nearby vehicles. However, such systems may fail to provide the location information if DSRC is not supported by one of the vehicles or in the case of a GPS unavailability and some of the shared data over DSRC or other networks could jeopardize user privacy. In this work, we focus on developing techniques that improve available location information and demonstrate the driver specificity of the shared data over such systems.

GPS is widely used in critical infrastructures but is vulnerable to radio frequency (RF) interference. A common source of interference are commercial drivers that use GPS jammers to circumvent vehicle tracking systems. Existing mechanisms to detect and identify such interference emitting vehicles on roadways require a large number of specialized detectors or a manual observation process. To detect GPS jammers on roads,

we designed a system that could detect any transmission at GPS L1 frequency. The key components of the system are monitoring stations (which are equipped with directional antennas and cameras) and mobile detectors (e.g., smartphones). Using an off-the-shelf software-defined radio (USRP) to emulate GPS jamming signals, we conducted a case study evaluation of our system with multiple trial drives on local highways in 2 US cities and found the monitoring stations effective. Through our experiments on a local highway with a vehicle transmitting interference in the 900MHz ISM band, we found that the vehicle identification rate of our mechanism is 65% for a single-point setup and 100% for a two-point setup.

To study the privacy of the data shared among vehicles, we designed a system that can access a rich set of in-vehicle sensor data through a custom CAN bus interface and examined its driver specificity. We designed classifier features that allow distinguishing drivers based on a minimal set of sensor data. We evaluated the system with data from 480 real-world trips collected over 3 weeks from five university mail vans, with 24 drivers in a controlled experiment, and 103 trips with four drivers across two households. Our system could achieve 91% accuracy within the 20s after the driver enters the vehicle in the real world experiments.

While the stream of rich sensor data can be communicated to and processed in a remote cloud, bandwidth and latency challenges encourage computation of this data on the vehicles themselves. With high computing powers and less power consumption, vehicles can sense the dynamic environment like no other platform. We propose to use harvesting vehicles as edge compute nodes, focusing on sensing and interpretation of traffic from live video streams. This work proposes effective fine-grained traffic volume estimation using in-vehicle dashboard mounted cameras. With the proposed system, we collect the footage of the traffic, detect vehicles using a real-time object detection method and estimate the lane of travel with the speed information for vehicles that are traveling in both directions. With such an information, not only the current positions of vehicles but also the estimated future positions of vehicles could be shown on a map. We conduct studies on different roads, our vehicle detection accuracy is over 75% even for highly occupied roads, and our speed estimation error is less than 12%. We could

also estimate the lane of travel with over 80% accuracy.

Acknowledgements

I would like to thank my adviser Marco Gruteser for his constant support and always leading me forward both academic and research world. Being his student is truly special. He has taught me to look at problems from multiple angles and look for best solutions possible. He will always have my sincere gratitude and best wishes.

I have learned a lot from my summer internship in General Motors, Michigan. I would like to thank Fan Bai who helped me to solve the research problems I encountered and infinite support for getting my first patent while writing a research paper.

I would like to thank WINLAB as a whole, including all the faculty and staff, who make WINLAB a great place for research. I enjoyed every day I spend there and the interaction among colleagues taught me many ways of tackling problems.

I also wish to thank National Science Foundation for monetary support of my projects and my collaborators and colleagues, including but not limited to: Ramesh Govindan, Yingying Chen, Ivan Seskar, Dipankar Raychaudhuri, Wade Trappe, Rich Martin, Yanyong Zhang, Yan Wang, Shubham Jain, Jian Liu, Jinzhu Chen, Tam Vu and Ashwin Ashok. I also would like to thank all my friends both inside and outside of Rutgers University.

Finally, I would like to thank my parents, Bulent and Faiker Kar; and my girlfriend Guner Giray for their constant support and love during these years. They always encourage me to follow my dreams, even it means we have to stay apart for many years. This dissertation is dedicated to them.

Table of Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

1.1 Overview

Each year, roads are shared by millions of vehicles and with this trend, this number will only increase in the future. In the coming years, with the deployment of DSRC systems vehicles are becoming smart, connected and increasingly autonomous. Vehicles would be broadcasting information such as the speed, heading, acceleration, and location multiple times per second so that each vehicle can be aware of nearby vehicles using these systems. With current plans, in the next few years, all vehicles would have built-in DSRC systems to enable safety applications. Although these systems could be installed into current vehicles, there would still be vehicles on roads that do not support DSRC systems. Detection of those vehicles would be a concern for connected vehicle systems.

The current implementation of DSRC systems based on GPS signals. If GPS is not available, safety systems cannot work properly. Also, if there are too much privacy concerns about the data to be shared in such systems, people would be reluctant to use such systems. In this work, we implement 3 individual components that can be a part of future connected vehicle system.

Due to the 20,200km distance to the transmitting satellites, the received signal strength of GPS systems is extremely low and therefore, they can easily be masked by other transmissions, either from jammers or malfunctioning electronic equipment. GPS jammers simply overpower GPS signals with interference. As a result of this, GPS jammers carried by vehicles in roadways is challenging and presents many obstacles that need to be solved. For example, the availability of GPS performance metrics. The GPS receivers only provide high-level GPS performance metrics that are not obvious indicators of interference. However, the equipment used is costly or at least not in widespread use. Both metrics, however, are also affected by the varying signal conditions under normal operation. Another reason this can be challenging is the vehicle identification in a complex environment. Roadways present both a complex radio propagation and traffic environment. The Doppler shift introduced by high traveling speeds, and unpredictable radio propagation paths due to the dynamic traffic patterns. All the irregularity make it challenging to accurately identify the vehicle that carries a GPS jammer. Standard localization methods are not sufficient to pinpoint a particular vehicle in dense traffic. And current methods require manual usage of handheld devices with a directional antenna to scan for signals. Therefore, an automated interference detection system is needed. This will be the first component of our system and will be discussed in the next section.

Modern automobiles contain hundreds of sensors and actuators that exchange data on internal buses. A small part of this data has already been exposed in the OBD-II standard but the majority has to date been used only internally. Recently, car makers have been experimenting with opening more of this information to smartphone or in-car apps. Such data is also increasingly accessible through telematics services and could potentially be processed in the cloud. One relevant question in this context is how driver-specific the data is. How easily can different drivers of a vehicle be distinguished from such in vehicle data or, more precisely, what is the minimal amount of data necessary to effectively distinguish drivers? The answer to this question will help in understanding the feasibility of building driver-specific applications for the many vehicles that are used by multiple drivers. We focus on shared vehicles in a professional or commercial setting, and personal use setting, where applications include personalization of vehicle settings (e.g., automatically adjusting entertainment, HVAC, transmission, or suspension configurations to driver preferences). Automated vehicle use logs, driver-dependent pay-asyou-drive insurance, or unauthorized vehicle use detection (where a vehicle might notify owners when it encounters an unexpected driver). The answer to this question will also contribute to an understanding of the privacy implications of such in-vehicle data. The more easily drivers can be distinguished, the less anonymous one can expect this data to be. This will be discussed in Section 3.

Vehicles have evolved from being mechanical systems to cyber physical systems, generating large amounts of real-time data. They offer high compute capabilities with far less power consumption concerns as other mobile platforms. Vehicles are powerhouses of energy, traversing through our physical world, and with the many sensors built in to them they are capable of sensing our dynamic environments like no other platform. We focus on harvesting vehicles as edge compute nodes, focusing on sensing and interpretation of traffic from live video streams. Vehicles are increasingly being installed with front facing cameras. Originally, these cameras were intended for a specific purpose such as lane detection and lane keeping. However, recently, with the trend in autonomous driving, dashboard mounted cameras have been used for applications ranging from simple pedestrian/car detection to enabling a self-driving system. In addition to enabling vehicle specific or driver specific services, these cameras can be leveraged for large-scale traffic analytics. Cameras in each vehicle have a unique perspective of the observed environment. For example, a car driving in the left most lane has a clear view of the cars driving in the same direction, as well as those in the opposite direction. Similarly, a car in the rightmost lane is optimally placed for detecting stalled vehicles. Each car can be enabled to process raw video stream and compute high level semantic information. Pre-processing raw video streams to extract high level information optimizes bandwidth usage, reduces latency, and conserves privacy. This information can then be shared with neighboring vehicles or a centrally located map service. The cloud-based map service can aggregate the information from a large number of vehicles to provide an up-to-date map of the region, overlaid with precise traffic and other road conditions information. This constitutes to a more accurate and sophisticated assessment of regions, compared to other approaches such as surveillance cameras, that do not cover all areas. This will be discussed in Section 4.

1.2 Organization

Chapter 2 presents the algorithms and experiments to detect GPS jammers on roads. Chapter 3 describes the user differentiation techniques we used and performance results.

Chapter 4 shows the traffic estimation techniques we used and performance results. Chapter 5 describes the related work. Chapter 6 concludes the dissertation with future research directions.

1.3 Contribution

This proposal presents the following contributions:

1. We proposed an automated two-pronged interference monitoring framework with roadside monitoring stations and mobile detectors that can achieve wide coverage at low cost. We designed profile-based algorithms that detect GPS signal anomalies based on widely available satellite signalto- noise ratio measurements. Our system can work with just a single detector (assuming repeated commuting by the jammer) to identify the interference source. We evaluated through multiple days of roadway experiments on local highways the likelihood that a monitoring station can uniquely identify an interference emitting vehicle. We also used our system to investigate the prevalence of GPS interference on US public roadways. During 200 hours of passive monitoring in two US cities, we captured two instances of interference.
2. We designed classifier features and a system that allow distinguishing drivers based on a minimal set of in-vehicle sensors data, with no additional hardware cost. We evaluated the system with data from 480 real-world trips collected over 3 weeks from five university mail vans as well as with 24 drivers in a controlled experiment. We found that data from the vehicle start is particularly specific to individual drivers, allowing our system to achieve 91% of accuracy within 20 seconds after the driver enters the vehicle in the real-world mail van experiment.
3. We used a real-time object detection technique to detect vehicles on roads. By matching SIFT features, we identified unique vehicles that are traveling in the opposite lane. By calibrating the camera, we estimated the lane each vehicle is driving and calculated the speed of the vehicles that are detected using our method and get less than %10 error in the speed estimation.

Chapter 2

Related Work

There are quite a few works analyzing the characteristics of the GPS jammers available in the market and utilizing multiple static stations to detect and localize the jammer. Bauernfeind et al. [13] proposes to mitigate the in-car jammers by applying an inverse signal transformation [60]. Mitch et al. [53] propose a time difference of arrival (TDOA) based method that can localize the GPS jammer with maximum error of 15 meters, which however requires the prior knowledge of the jamming signal. Lindstrom et al. [50] builds a network of low cost application-specific integrated circuit (ASIC) front-end modules providing automatic gain control (AGC) values to detect and localize the interference sources in GNSS L1/E1 band with an error of 21 meters on average. Similarly, Poncelet et al. [63, 16] can detect and localize wide-band GNSS jammers based on TDOA measurements with the accuracy of 20 meters.

Vehicular ad hoc networks are also being employed for identifying and localizing GPS jammers. Bauernfeind et al. [12] uses a Differences-of-Received-Signal-Strength (DRSS)-based localization algorithm which requires at least signal measurements from four vehicles for localization. Fontanella et al.[32] further experimentally confirms the potential for using VENETs to locate in-car jammers. However, the assumption of having vehicles all around the jammer is not generally true and the estimation error is approximately 40 meters which is too high for automatic vehicle identification. In addition, Kramer et al. [48] recently proposes an Android based approach which relies on relative position change between a smartphone user and a static GPS jammer to perform localization. Scott et al. [70] proposes a crowd-sourcing approach to detect and localize the GPS jammer by exploiting the AGC values from the GPS receiver, which requires to incorporate the GPS jam-to-noise (J/N) ratio detector in cell phones.

Chronos Technology has two commercial devices [19][20] for detecting and localizing GPS jammers in the L1 frequency band. However, these devices can only detect jammer in static scenarios and can only be used manually for a short period of time. On the contrary, our solution is automatic, can be deployed at roadsides and can detect jammer in dynamic mobile scenarios.

In the realm of driver distinction, Choi et. al [18] show that driver behavior can be modeled using steering angle, brake status, acceleration status and vehicle's speed that are collected from vehicle's CAN bus. The driver identification part resembles our methodology however since they are only using longitudinal data collection, the accuracy of driver identification is less than 35% with their system. In another driver distinction work, Enev et. al [28] could manage to identify 15 drivers with 100% accuracy using their longitudinal behaviors. Their system needs at least 15 minutes of training dataset for each user, that includes braking patterns, vehicle speed, acceleration, throttle position etc.

In another work, Miyajama et. al [54] show that up to 276 drivers could be identified with 76.8% accuracy using gas/brake pedal usage, engine speed, steering wheel angle and car following distances. Their system works reasonably well for 276 drivers using longitudinal behaviors but their system also needs 5 minutes of training dataset for each user. With our proposed design, we can attain up to 95% accuracy in under 30 seconds.

Driver identification is also investigated by Riener et. al [66]. The authors used sitting postures to distinguish drivers using a pressure pad. However, privacy of the collected data is of utmost importance. And while a fair amount of related work proposes to use speed and location coordinates for privacy preservation, Krumm et.al [49] have proved that location traces can be used to successfully identify individuals. Gao et. al [33] have also demonstrated that speed is not a privacy preserving parameter and is enough to track a driver.

Researchers have been exploring ways to maintain driver privacy and anonymity, by masking identifying data. Hoh et. al [39] have proposed using virtual trip lines to maintain driver privacy. In another work, Zan et.al [80] guarantee a high degree of anonymity by using a zone-aware path cloaking scheme. Another approach has been investigated by researchers in [21], [22], [67] and [68] with the usage of a Trusted Third Party to perform cryptographic operations.

Modeling and predicting human behavior has been investigated in another work by Pentland et.al [62]. The authors achieved 95% accuracy at predicting automobile driver's actions from their initial preparatory movements. However, the algorithm can only determine when a car will be passing another, turning or following the previous cars in next couple of seconds. There has been much work on systems for traffic monitoring, rather than driver monitoring both in commercial companies and research facilities. Many of them leverage GPS units on cars(OnStar[5] system) to track the vehicle's movements and analysis can be done at the server. The Nericell[56] project concentrates on the road topology and shows that potholes, bumps and braking can be detected by using accelerometer and GPS sensors of the mobile phones. In another work [72], the authors show how driver's behavior under critical circumstances(sudden breaks, extreme steering angle rotations) varies compared to regular times, using smartphones. In contrast to smartphone sensor based techniques, we use vehicle sensors for more reliable measurements.

Chapter 3

DETECTION OF ON-ROAD VEHICLES EMANATING GPS INTERFERENCE

3.1 Overview of Chapter

Due to the 20,200km distance to the transmitting satellites, the received signal strength of GPS systems is extremely low and therefore, they can easily be masked by other transmissions, either from jammers or malfunctioning electronic equipment. In the present study, we seek to overcome the challenges of detecting a moving vehicle that carries a GPS jammer, with an automated interference detection and vehicle identification system that includes roadside interference monitors, mobile interference detectors and camera-based vehicle identification, as shown in Figure 3.1.

The roadside component of such a system could remain in continuous operation near critical infrastructure and thereby quickly identify interference in such areas. The mobile detectors that leverage smartphones or other mobile devices opportunistically to identify interference can achieve wide coverage at low overhead. It requires a special profiling approach, however, to distinguish interference from benign fading and attenuation based on the limited GPS performance indicators available on widely used mobile devices. Our approach combines data from multiple detections to identify vehicles. The inherent variation of driving speeds means that a group of cars that passes a first monitoring point together likely has dispersed at a later monitoring point, which allows unique identification of the vehicle emitting such signals.

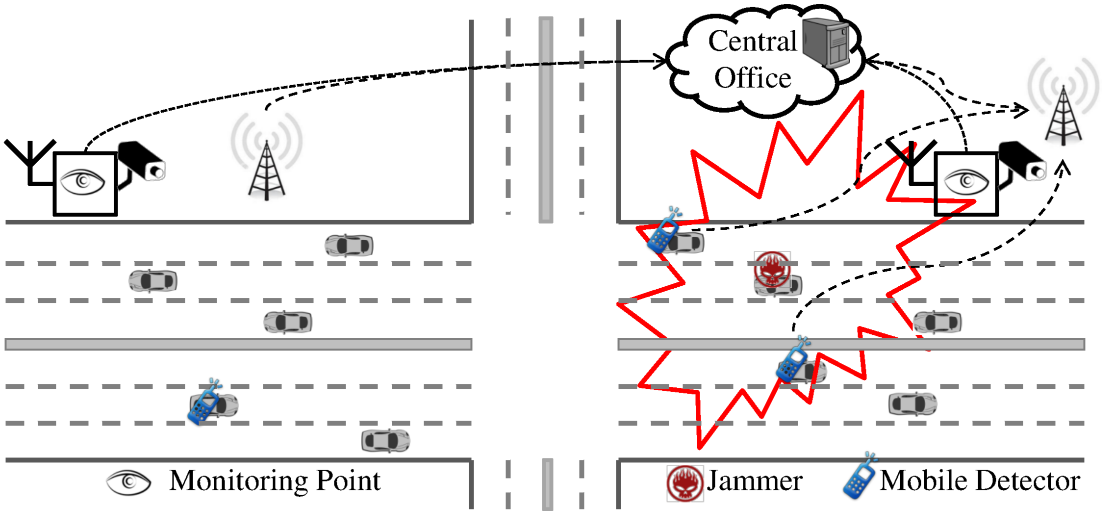
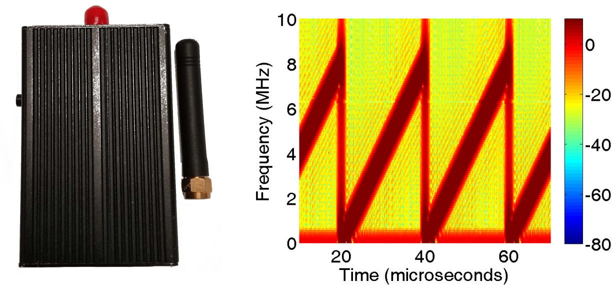


Figure 3.1: Overview of jamming detection system consisting of monitoring points, mobile detectors, and central office for camera-based vehicle identification.



(a) GPS jammer (b) Spectrogram of a jamming signal

Figure 3.2: (a) shows one of the commonly available jammer and (b) shows the spectrogram of a 8 MHz wide jamming signal generated using Matlab function.

3.2 System Overview

Our proposed detection system uses unmanned roadside monitoring stations in critical locations (e.g., airports) and crowdsourced GPS signal-to-noise ratio data from mobile phones or vehicles to achieve broad coverage. It is designed to significantly reduce the effort required to detect and track down interference in the civilian use L1 GPS carrier at 1.57542 GHz. We envision that such a system can be used for identifying malfunctioning equipment that generates interference or for enforcement actions against GPS jammers.

3.2.1 Attack Model: Personal Privacy GPS Jammers

The attackers are mainly drivers that want to gain location privacy by circumventing employer-installed GPS vehicle tracking systems with GPS jammers. We target relatively widespread low-cost jammers that are used as personal privacy devices rather than sophisticated defense-related equipment. Such personal privacy devices are often small jammers that can be powered by an automotive power outlet.

These GPS jammers simply overpower GPS signals with interference. Although some advanced jammers can block the L1, L2, L3, L4 and L5 bands, the most commonly used ones block only the L1 band. Such GPS jammers can be categorized into two categories in terms of signal characteristics [12]: (a) Continuous wave jammers simply transmit an unmodulated continuous wave signal with a bandwidth less than 100 kHz [64], and (b) Chirp jammers transmit different types of narrowband and wideband chirp signals where the chirp signal can be a saw-tooth function, a multiple saw-tooth function, or a frequency bursts.

The jamming bandwidth ranges from 0.92 kHz to 44.9 MHz and the output power varies from -9.5 dBm to -30.8 dBm [13]. Chirp signal jammers are most common [13]—we show an example in Figure 3.2(a). For our experiments and analysis, we generated a chirp signals similar to the characteristics reported by Bauernfeind et al [13]. An 8 MHz chirp signal is depicted in Figure 3.2(b).

3.2.2 Design Requirements

Towards this goal, we identify the following design requirements for identifying vehicles emanating GPS interference.

- **Automated.** The detection and identification system should be able to operate around the clock with limited human intervention. While manual examination of detection results may still be required, continuous human presence at a monitoring

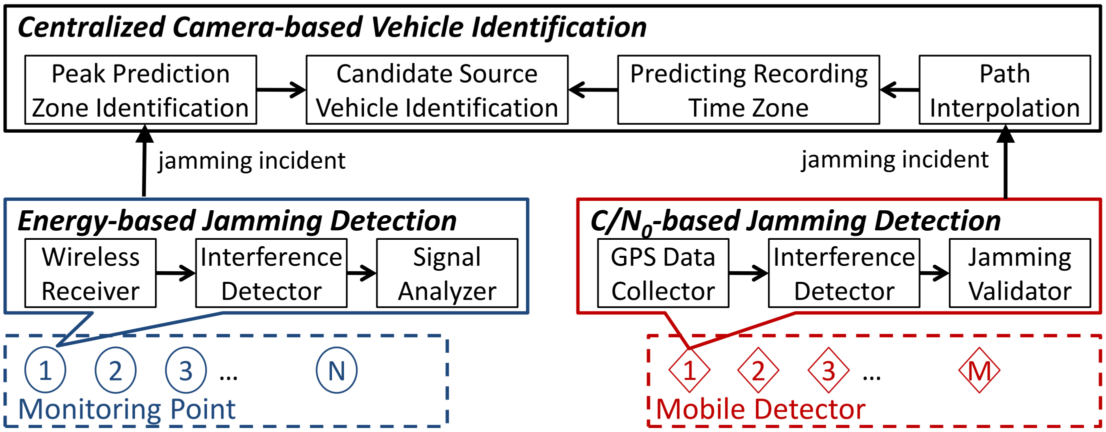

Centralized Camera-based Vehicle Identification
Peak Prediction Zone Identification
Candidate Source Vehicle Identification
Predicting Recording Time Zone
Path Interpolation
jamming incident
jamming incident
Energy-based Jamming Detection
Wireless Receiver
Interference Detector
Signal Analyzer
C/N0-based Jamming Detection
GPS Data Collector
Interference Detector
Jamming Validator
1
2
3
…
N
Monitoring Point
1
2
3
…
M
Mobile Detector

Figure 3.3: Overview of our GPS jamming detection methodology. Each monitoring point uses a USRP for capturing wireless data and each mobile detector uses phone's GPS APIs to collect SNR values. The centralized camera-based vehicle identification system uses traffic camera images for accurate vehicle identification.

location should not be necessary.

- **Low-cost.** To facilitate widespread usage, infrastructure costs should be minimal. Some infrastructure is appropriate for priority locations around critical GPS infrastructure (e.g., airports).
- **High accuracy.** Identifying the source of the interference source should be accurate enough to support enforcement actions. False positive can lead to inconveniencing innocent drivers and false negative represents missed detections.

3.2.3 Challenges

Identifying GPS jammers carried by vehicles in roadways is challenging for several reasons.

Availability of GPS performance metrics. Widely used GPS receivers only provide high-level GPS performance metrics that are not obvious indicators of interference. Identification of interference in the GPS L1 band is not particularly difficult with specialized equipment or high performance GPS receivers that monitor interference conditions. Such equipment, however, is costly or at least not in wide-spread use. Many GPS receivers only allow monitoring satellite signal-to-noise ratios (SNR) and outages (i.e., a loss of the position fix). Both metrics, however, are also affected by the varying signal conditions under normal operation.

Vehicle identification in complex environment. Roadways present both a complex radio propagation and traffic environment. The radio propagation on a multi-lane highway is highly irregular because of the strong attenuation and fading caused by the metal bodies of vehicles, the Doppler shift introduced by high traveling speeds, and unpredictable radio propagation paths due to the dynamic traffic patterns. All the irregularity make it challenging to accurately identify the vehicle that carries a GPS jammer.

Several projects have proposed to localize GPS jammers, but none of them can automatically identify vehicles with GPS jammers accurately and economically. For instance, most of the latest research on GPS jammer localization are based on the time difference of arrival (TDOA) technique [50], [53], [63]. However, TDOA-based approach requires either high accurate synchronization between multiple observing stations or the prior knowledge of GPS jamming signals. In addition, these localization mechanisms are not accurate enough to identify a single vehicle. While some of them can localize the source of jamming signals with an accuracy of 5 meters in certain cases, this is still not sufficient to distinguish two vehicles driving on adjacent lanes. Thus, a new vehicle identification solution is in need.

3.2.4 Approach

We chose a two-pronged approach with specialized road-side detectors and mobile detectors because the former promise higher system accuracy around critical infrastructure and the latter offer low-cost, broad coverage of an entire transportation network. As shown in Figure 3.3, both types of detectors will report signal anomalies to a centralized system. Rather than attempting precise localization at a single point in time, the system uses multiple reports to identify the source vehicle. To do so, it must have access to vehicle identification information such as electronic toll system identifiers or camera footage that reveals license plates. We focus here on the latter as an example.

For specialized roadside equipment, there exist established detection methods. In the interest of low complexity, we rely on energy-based detection, wherein a `Wireless Receiver` continuously monitors the GPS L1 frequency band and the `Interference Detector` monitors whether energy exceeds the normal noise levels at this location. Suspicious signals can further be validated against established jamming patterns by the `Signal Analyzer` and are reported to the centralized system together with their location.

Interference detection is most challenging when seeking to collect interference data from widely deployed, mobile GPS receivers. We chose to focus on smartphone GPS receivers accessible through Android apps, because this is arguably the most common GPS platform. For example, such mobile detectors could be carried in government vehicle fleets or with appropriate energy optimizations offered as an Android background service to the public. The Android platform makes available carrier-to-noise density ratio—loosely called signal-to-noise (SNR)—measurements of each satellite in addition to the GPS position data, but does not provide any information that would directly indicate interference. Interference, however, can be expected to manifest itself as additional noise at the receiver and therefore decrease the carrier to noise density ratio (or, a loss of fix, in the extreme). The challenges lie in distinguishing this effect from signal fading or attenuation, which is prevalent in wireless propagation environments and also leads to carrier to noise density ratio decreases or outages.

Our mobile detection approach therefore profiles the environment to establish reference signal levels for a certain receiver. Signals are then classified as anomalies, when they deviate significantly from this profile. They can be further validated by detecting abnormal patterns in carrier-to-noise-density ratio variations. These three steps make up the `GPS Data Collector`, `Interference Detector`, and `Jamming Validator` components. We discuss detection in more detail in Section 3.3.

The centralized camera-based vehicle identification component identifies a vehicle using images from roadside units or traffic cameras. For roadside detection, it is most convenient to have cameras collocated with the interference detector. Still, in many cases one can expect multiple vehicles to be present in the image and it will be difficult to identify which one is the interference source. We address this challenge by relying on multiple detections in time or space. Since groups of vehicles will eventually disperse, multiple detections (of the same jammer) will eventually show only one common vehicle. When mobile detections are reported, there may not be any camera available at this location. In this case, the system will have to interpolate the vehicles path to a camera location, and work with a larger set of candidate vehicles to account for the path estimation uncertainties. Still, with a sufficient number of detections, unique identification can be expected. We discuss this component in Section 3.4.

3.3 Jamming Detection

While many sophisticated jamming detection solution exist, we focus here on techniques that can be implemented with existing widely available GPS receiver hardware or low cost software radios. In particular, we develop a carrier-to-noise profiling approach which only relies on the carrier-to-noise metrics (sometimes imprecisely referred to as signal-to-noise ratio) reported by most GPS receivers. This metric is even accessible for apps through the Android location API. We also consider energy detection which can easily be implemented on dedicated monitoring devices and is also supported in more sophisticated GPS receivers. While not nearly as widely available as the carrier-to-noise metric, energy detection allows a more direct measure of interference since it does not need to account for variations in satellite signal propagation.

3.3.1 Carrier-to-Noise Profiling

To achieve broad coverage with this monitoring system, we design a solution that can take advantage of existing GPS chips and devices such as smartphones or in-car GPS receivers connected to telematics units. The carrier-to-noise density ratio (C/N_0) is the most fine-grained GPS performance measure that conveys information about interference and is provided by nearly every GPS chip as well as made accessible through the NMEA standard or through the Android API. It is the ratio of the satellite signal power to noise density and is often somewhat imprecisely referred to as signal-to-noise ratio (SNR) in many GPS interface documentations.[1]

The C/N_0 is primarily affected by signal variations which are due to fading, both from obstructions (e.g., skyscrapers, tall billboards), satellite movement, and multipath propagation. Under interference conditions, however, the interfering signal acts similar to noise on the satellite signal reception process. The reported C/N_0 is therefore usually a signal-to-interference-plus-noise density ratio, which will decrease when either interference increases or signal fading occurs.

The challenge is then to distinguish fading from interference. We propose to detect abnormally low SNR values by comparing them with SNR profiles that characterize typical fading variations at a location. We rely here on three key observations: (i) in-vehicle jammers and interferers are mobile and rarely dwell for an extended amount of time in the same location (indeed, they are usually deactivated when a vehicle parks); (ii) vehicles pathways are highly constrained by roadways and tend to repeatedly pass through those; (iii) most significant signal fading is cause by stationary structures such as tall buildings, underpasses, tunnels, etc. The first observation leads us to the heuristic that a temporarily low SNR at a particular location is indicative of interference, while a more permanently low SNR at a particular location is more likely due to obstructions in the built environment. The second and third observations leads us to the idea to construct a profile of typical signal levels at each location. If detectors are mobile, each small interval of length d along a roadway is treated as a separate location, which will eventually lead to a signal map. By comparing current SNR readings with this profile, rather than a global threshold, we can then perform outlier detection to determine whether the readings are unusually low for this particular location. This should significantly reduce false positives. Thus, our algorithm contains two stages: profiling, and jamming detection. The generalized mobile version is shown in Algorithm 1 and we describe the individual stages next.

Profiling. In this stage, we first collect C/N_0 values along a route from the n

---

[1] In wireless communications, SNR typically includes noise over the entire receiver bandwidth, while C/N_0 uses noise density, that is noise over 1 Hz. We use the terms interchangeably here to refer to the carrier-to-noise ratio density.

strongest satellites for each of the position readings along the route. Note that as satellites orbit the earth, the identity of these satellites will change but we found C/N_0 readings to be sufficiently consistent to not warrant further distinction. For each interval p_i along a roadway, we calculate and store the mean m, the mode τ, and the standard deviation σ of the C/N_0 values from all satellites and all position updates within this interval. By calculating a mean over multiple C/N_0 readings (from different satellites but also different locations within the interval or different trips), we will average out multipath effects and the approach becomes more robust to small scale fading. The mode is provided as a statistic that is more robust to outliers. We chose 20m as the interval length d, because it is sufficiently fine-grained to capture signal variations cause by smaller structures, and large enough so that usually at least one sample from a trip falls into this interval. These values comprise the profile and can be continuously updated as additional C/N_0 data from the same location arrives (i.e., the vehicle travels over the same road again).

Anomaly Detection. In this stage, we use the parameter recorded in the profiling stage to determine whether the SNR at a particular location is abnormally low. First, the current reported latitude and longitude are mapped to a particular road interval p_i. Second, we will again calculate the mean SNR m_{snr} over all satellites and all location updates from the current trip that fall into this interval. This is reasonable even if we expect interference, because (i) interference will affect all satellites equally since they are transmitting on the same L1 carrier and (ii) in most cases interference will be present long enough for the vehicle to travel a short distance d (e.g., 20m). Thus, the averaging is unlikely to remove the effect of interference.

The mean of the profile and the sigma of the profile are then used to derive a threshold for outlier detection. An abnormal signal report is generated if $m_{snr} < m_{pr} - k * \sigma_{pr}$. Since the normal distribution is often successfully used to model the distribution of SNR readings one might expect at a given distance, it can serve also as a guide for determining the k parameter of the threshold. While this is not necessary for built-in vehicle GPS receivers with external antennas, the threshold for portable GPS receivers can be further calibrated to account for placement differences inside the vehicle from one trip to the next. For this purpose, the algorithm will also track the mode of the SNR τ_{snr} over an extended distance d_{ext}, which is chosen longer than the longest expected interference duration. The algorithm then estimates the profile mode over the same extended interval based on the m_{pr} that make up this interval. The difference between this estimated profile mode and the current measured mode is a calibration parameter that will be added to m_{snr} before the comparison.

When the SNR falls below this threshold, a signal anomaly report can be transmitted to a central office. This report will contain the difference between the threshold and the mean SNR value as well as the current or the last known GPS location, if the current one is unavailable. In the latter case, the first new location fix may also be provided subsequently, to allow a better estimation of the location of the low signal. Complete outages, where no satellites can be detected, are treated as the lowest possible SNR value in the algorithm. The low signal report can further include indicators of detected signal characteristics which imply a higher confidence that this signal level is due to jamming, such as the high variance described earlier. The central office can take the magnitude of the SNR difference and such indicators into account, when determining the reliability of such reports.

3.3.2 Comparison with Energy-based Detection

Roadside units designed for interference monitoring can rely on many custom techniques to detect interference. Energy detection is a simple yet effective technique in the GPS band, since the GPS signal level is typically below the thermal noise floor (confirmed in our experiments). Therefore, any energy-level above the noise floor is unwanted interference. This approach greatly simplifies the choice of detection threshold since no signal variations have to be taken into account but it requires hardware that supports such metrics.

Based on these considerations, we chose the carrier-to-noise-profiling approach in our mobile smartphone detectors and implemented energy detection in our dedicated roadside detector using a Universal Software Radio Peripheral (USRP) N210 [29] in

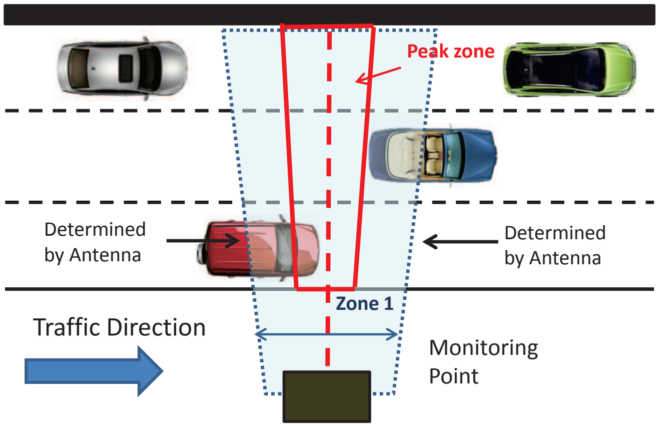


Figure 3.4: Ladder-shaped detection zone on the road.

conjunction with a WBX daughterboard and with GNURADIO [36] scripts. The roadside unit first automatically sets the threshold value just above the ambient noise level. For monitoring, the `wireless receiver` block (Figure 3.3) continuously captures 10 milliseconds of data and passes it to `interference detector` block. When the energy becomes larger than the threshold ($-75dBm$ to $-83dBm$ in our experiments), the interference detector records a suspicious signal and reports it together with timestamp information. A `signal analyzer` can optionally conduct temporal and spectral analysis to gain additional confidence in the report.

3.4 Source Vehicle Identification

In this section, we describe how to correlate the detection with license plates from traffic camera footage to allow identification of a set of candidate vehicles and how this set can be narrowed down to a unique vehicle after multiple detections. While we discuss this in the context of traffic cameras, similar methods could be applied to other vehicle identification methods such as IDs from electronic toll collection systems.

3.4.1 Identification Using Roadside Detector

To facilitate vehicle identification, the system needs to map an energy detection to a portion of a video obtained from a traffic camera that the monitoring point is collocated with. It is therefore convenient to focus the wireless detections on a smaller area that is in view of the camera. This can be achieved by using a directional antenna at the monitoring point and results in a trapezoidal area of the roadway from which signals can be detected. Such a detection zone is illustrated in Figure 3.4 (marked as *Zone 1*). Let us first consider a naive approach to identify a set of candidate vehicles. Given a time interval $[t_s, t_e]$ during which interference was detected, consider every vehicle that was sighted in the monitoring zone at least at one point of this time interval. In practice, however it is challenging to determine this zone since it not only depends on wireless propagation effects but also the transmission power and antenna of a jammer.

Deriving Peak Detection Zone. To address this we use the concept of a peak detection zone. As an interfering vehicle approaches and leaves, the received signals at the monitoring point can be expected to rise and fall, respectively. By considering only the moment t_m at which the energy of interference signals peaks, we can define a smaller peak detection zone.

Theoretically the energy of interference signals reaches the maximum value when the transmitting vehicle is at the closest position to the detector. These closest positions are along a line perpendicular to the road and passing through the detector (red dotted line in Figure 3.4). Due to multipath such as reflections of interference signals by other vehicle bodies, there is however some uncertainty regarding the exact position of the vehicle when energy peaks. We therefore use a *peak detection zone* as the area that includes all those possible positions (i.e., the red trapezoid in Figure 3.4).

This discussion implicitly assumed omnidirectional transmitters. For a transmitter that is using a directional antenna, the peak may in theory not occur when the transmitting vehicle is closest to the detector but most commercially available jammer in the market have omnidirectional antennas and in practice, the tolerance zone can compensate many discrepancies.

Obtaining the boundary of the peak detection zone theoretically is complicated and thus we identify the zone empirically. We use the following steps: 1) For each experiment, we identify the position (p_m) of the transmitting vehicle at the moment t_m by examining the recorded video; 2) draw a ladder-shaped area to cover all the marked

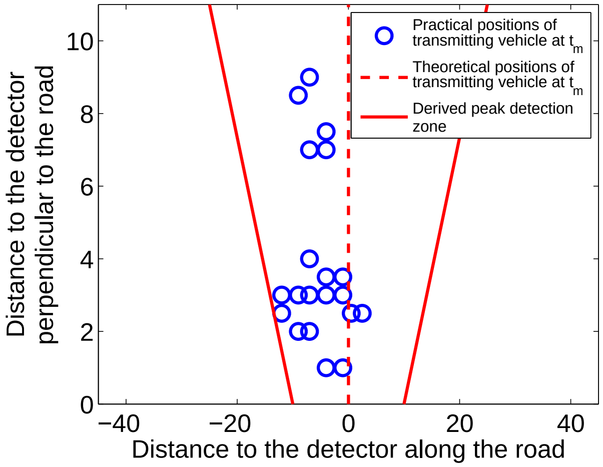


Figure 3.5: Positions of transmitting vehicle passing by the detector at time t_m when the maximum energy value is identified in multiple experimental runs in real-road scenarios.

vehicle positions (p_m) across all the lanes. Figure 3.5 shows the identified vehicle positions at the moment t_m for the 20 rounds of real-road experiments. Empirically, we determine that the peak detection zone for our receiver configuration can be enclosed by an trapezoid with a width of about 15 meters. Overall, the results are encouraging as it indicates it is possible to derive a peak detection zone that minimizes the number of candidate vehicles to uniquely identify the source vehicle.

3.4.2 Identification Using Mobile Detector

When multiple mobile devices (e.g., smartphones) on the road detecting the presence of the GPS jamming/interference, our system can track and predict the movement of the jammer on the road and coordinate with the roadside cameras to capture the candidate interference vehicle.

Determining the Moving Direction of Jamming /Interference Source. When the GPS jamming /interference is detected by a mobile detector (e.g., a smartphone), it is possible to determine the moving direction of the vehicle emanating interference by utilizing the *detection period*, which is the time period that the mobile detector finds low GPS SNR (i.e., below the detection threshold). The rationale behind this is if the mobile detector moves in the opposite direction as the jammer, the detection period will be very short, typically 1-2s. Whereas if the mobile detector moves towards the same direction as the jammer, the detection period will be longer than few seconds. Thus, by utilizing a time threshold we can determine the relative direction of the GPS jamming source.

Intuitively, the mobile detector moving in the opposite direction as the source vehicle is the most common case, since a vehicle has much more chances to encounter different vehicles in the opposite direction than in the same direction. Therefore, we focus on the cases when the moving direction of the source vehicle is opposite to that of the mobile detector. Since the source vehicle is moving along the road, it is reasonable that there are multiple mobile detectors sequentially detecting the presence of jamming at several different locations on the road. It is thus possible to determine the speeds of the source vehicle and its estimated locations, and further perform the path interpolation to estimate the time when the source vehicle passes the closest installed roadside camera.

To estimate the number of mobile detectors needed to guarantee at least one detector-jammer encounter, we built an emulation to analyze an actual dataset, called Next Generation SIMulation (NGSIM) [42]. The dataset contains actual trajectories of more than 2100 vehicles recorded by video cameras mounted on the side of an urban highway in North Hollywood, CA during a 15min rush hour period. The emulation randomly selects one vehicle to be a jammer and M vehicles to be mobile detectors. For each selection, the emulation looks at the vehicles' traveling timestamps and trajectories to detect if any mobile detector encounters the jammer. The emulator repeats 10,000 times for each value of M ranging between 1 and 10. The results shows that the jammer can be encountered 92% of the time with 10 mobile detectors. The number reduces to 91%, 83%, 74%, 67%, and 51% when the number of monitoring points reduces to 9, 5, 3, 2 and 1 respectively.

Estimating Recording Time Zone.

Next, we first present how to estimate the speed of the source vehicle based on two GPS signal SNRs observations from a single mobile detector at two time points. Then we discuss how to determine the recording time zone by leveraging the estimated speed and location of the source vehicle.

Assuming that at time t_1 the mobile detector detects the presence of GPS jamming on a road with the opposite direction, the detector reports its speed v_m, the maximum SNRs of the GPS signal before the detection S^0 and after the detection S^1 to a centralized server. Based on S^0 and S^1, the central server can calculate the noise power after the detection as $N_1 = S^0 - S^1 + N_0$. Then it can obtain the jamming signal's amplitude by subtracting the thermal noise from the noise after the detection, and further estimate the jamming signal power by taking the square of the jamming signal's amplitude using the equation:

$$\hat{J}_1 = 10 \log_{10} (\sqrt{10^{N_1/10}} - \sqrt{10^{N_0/10}})^2, \tag{3.1}$$

where N_0 can be calculated with room temperature and signal bandwidth [71]. Note that the $\hat{J}_1$ in Equation (3.1) is converted to decibel unit. Since most civilian GPS jammers have similar jamming power [13], the central server can estimate the relative distance d_1 between the jammer and the detector with the estimated jamming signal power $\hat{J}_1$ based on the free-space path loss model[61]. Similarly, the central server can estimate the relative distance $d_2 = d_1 - \Delta d$ at time $t_2 = t_1 + \Delta t$ by using the S^0 and another maximum SNR S^2 reported by the detector at time t_2 (within the detection period). The estimated speed of the source vehicle $\hat{v}_J$ can be calculated as following:

$$\hat{v}_J = (\Delta d)/(\Delta t) + v_m, \tag{3.2}$$

The estimated speed of the source vehicle keeps updating whenever there are new smartphones reporting the detection of the jamming/interference. Assuming the individual vehicle speeds follow a invariant normal distribution [57], the central server can fit the $\hat{v}_J$ in Equation (3.2) to a normal distribution $\mathcal{N}(\mu_v, \delta_v)$, where μ_v and δ_v are the mean and standard deviation of the speed, respectively. Since the central server can also estimate/track the locations of the source vehicle by utilizing the last-fixed locations reported by mobile detectors on the road, when the central server finds that the source vehicle will pass a roadside camera that is at a distance of d_c away, it can estimate the traveling time of the vehicle to reach the camera as:

$$\hat{t}_c = (d_c)/(\mu_v), \tag{3.3}$$

and start recording on the camera around $\hat{t}_c$ to capture the candidate vehicles. We define the *recording time zone* as $\{\hat{t}_c - \theta_1, \hat{t}_c + \theta_2\}$, where θ_1 and θ_2 are adjustable parameters capturing the estimation errors, which can be calculated as following in our work:

$$\begin{aligned} \theta_1 &= (d_c \cdot 3\delta_v)/(\mu_v(\mu_v + 3\delta_v)), \\ \theta_2 &= (d_c \cdot 3\delta_v)/(\mu_v(\mu_v - 3\delta_v)). \end{aligned} \tag{3.4}$$

To get an understanding of the relationship between θ and the distance from the interference observation, we analyze the difference in travel time of vehicles on a common road segment using the NGSIM dataset mentioned above. The emulation randomly selects 2000 pairs of intervals that are 100 to 500m long on this road segment. It then calculates the amount of time it takes for each vehicle to travel through this interval. Figures 3.15 shows the distribution of travel time for each interval length, in 100 meter increments. The spread in travel times is directly related to theta. For example, the results show vehicles took between 25 to 55s to travel the 500 meters intervals. This means that if the camera is 500 meters away from the mobile detection, an appropriate size for θ_1 and θ_2 would be 15s, which together results in the 30s spread in travel times observed.

3.4.3 Monitoring Point Vehicle Identification

The formulation of the peak detection zone and recording time zone equip our system to derive a new approach that can automatically identify the transmitting vehicle by exploiting the camera visualization. When a single monitoring point with a roadside detector and a monitoring camera is available, the moment t_m when the detector finds the interference signals having the maximum value is utilized to check the video captured by the monitoring camera. And the peak detection zone is applied to capture all the candidate vehicles. We show in the next Section that we find over 95% of our experiments in real-road scenarios capture only one candidate vehicle inside the peak detection zone.

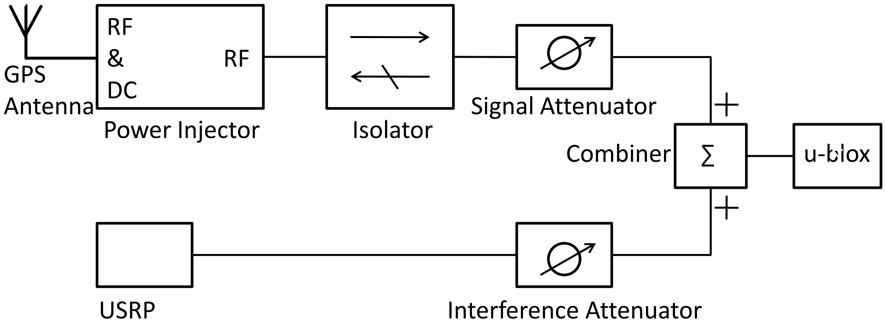


Figure 3.6: The wired experiment circuit.

In low traffic conditions, using single monitoring point can identify transmitting vehicles with high confidence, because in most of the cases there is only one transmitting vehicle in both the peak detection zone and the recording time zone. However, if the traffic is heavy, multiple candidate vehicles could be captured by either the peak detection zone or the recording time zone. By using our low-cost solution, our system can employ multiple monitoring points separated along the major roads having high traffic volume to still uniquely identify the transmitting vehicle. The intuition behind this is that with two or more monitoring points along the same direction of the major roads, it is less likely for a transmitting vehicle to have the exact same neighboring vehicles inside of either the peak detection zone or the recording time zone when crossing different monitoring points.

Thus, the multiple monitoring points identification algorithm is separated into 2 steps:

Step 1: Collect Images of Candidate Vehicles from Monitoring Points. Assume the transmitting vehicle passing K monitoring points on the road, the cameras at these monitoring points are all triggered to record the videos when the transmitting vehicle passing by. Therefore, each monitoring point generates the images of candidate vehicles (including the transmitting vehicle) from the recorded videos, denoted as $R_k = \{r_1^k, \ldots, r_m^k\}$, where $k = 1, \ldots, K$ is the index of monitoring points, r_m^k is the images of m^{th} candidate vehicles captured at the k^{th} monitoring point. All R_k are transmitted to a central server. We note that this step is also valid if the transmitting vehicle passing the same monitoring point for K times in one day or different days, similarly, R_k from the same monitoring point will be transmitted to the central server.

Step 2: Cross Validate Candidate Vehicles. Once the central server receives the sets of vehicle's image R_k from all monitoring point cameras, it finds the common vehicle C that appears in all R_k by calculating the $C = R_1 \cap R_2 \cap \ldots \cap R_K$.

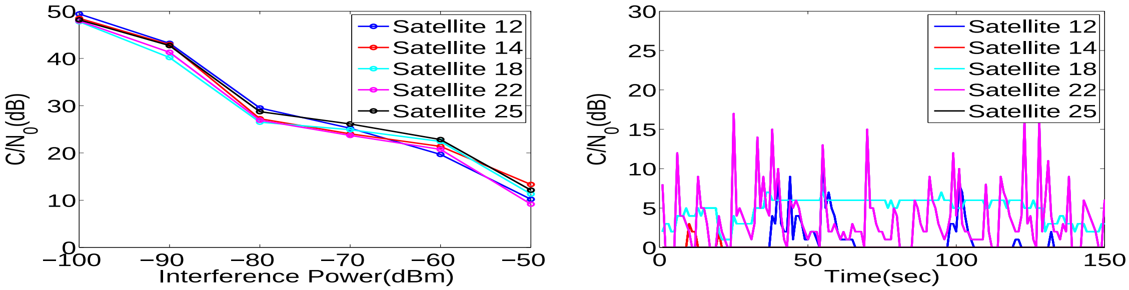
3.5 Performance Evaluation

In this section, we evaluate the two most important components in the designed system: Carrier-to-Noise (C/N$_0$) profiling based detection in both static and mobile cases, and source vehicle identification leveraging energy detection through roadside monitoring. Such performance evaluation provides fundamental insights to a complete system implementation for GPS interference detection.

3.5.1 Stationary Carrier-to-Noise Profiling and Interference Tests

To gain a deeper understanding of the effect of an jamming signal on SNR levels of GPS receivers, we conducted the following experiments. To comply with federal regulations and to minimize possible interference to GPS systems, we created a wired testbed depicted in Figure 3.6. In this setup, a jamming signal created by a USRP N210 software defined radio is combined with GPS satellite signals from an external antenna and fed into a u-blox EVK-7N GPS receiver [73]. We used an isolator to ensure that the jamming signal is not transmitted by the GPS antenna. In addition, we utilized a high gain (40dB) GPS antenna with attenuators to adjust the SNR of the received GPS signals. Using this setup, we first analyze the SNR patterns of different satellites with a 10 hour experiment. In this test, we did not use any attenuation to be able to observe the natural SNR behaviour of satellites. The result of our detection algorithm is depicted in Table 1 as the stationary case. Then by using this setup, we analyze the impacts of interference power on different satellites. As shown in Figure 3.7(a), increasing interference power results in a linear decrease of SNR with increased noise. This result verifies the expected effect of interference on the SNR readings.

We also observe that when a GPS receiver loses a location fix due to high-energy interference it sometimes still can receive data from some satellites, at least for part



(a)

(b)

Figure 3.7: Result of cable experiments where (a) shows C/N_0 levels of satellites for different interference power levels, and (b) shows C/N_0 levels of satellites when GPS receiver lost location fix.

of the time. We show the SNR level of several satellites over time at an interference level of $-45dBm$ in Figure 3.7(b) where the GPS receiver has lost the location fix. We can observe high variance in several satellites under this high level of interference, an effect that we have not observed under any of our non-interference tests—both in this wired setup and the vehicle experiments described later. This pattern can therefore be a further indicator for interference.

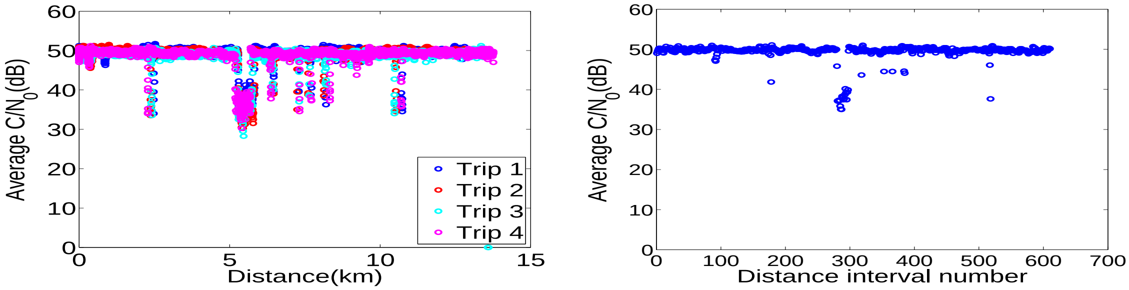
3.5.2 Mobile Detectors for Carrier-to-Noise Profiling

We study the performance of mobile detectors leveraging mobile devices (e.g., smartphone and GPS receiver) placed inside of vehicles. Specifically, we examine the detection rate and false positive rate under various interference powers and the behavior of SNR changes across different mobile devices over repeated road trips.

Experimental Setup

Hardware and Signals. We conduct our experiments in two routes in two US cities. In the first city, we use 1 HTC Evo 4G phone, 1 LG Nexus 4 phone, and 1 u-blox EVK-7N GPS receiver. We use 1 LG Nexus 4 phone in the second city. During the experiments, smartphones are located on the front panel of the car and GPS antenna for u-blox receiver is placed on top of the car.

Real Road Scenarios. We chose a suburban highway for route 1 with a distance



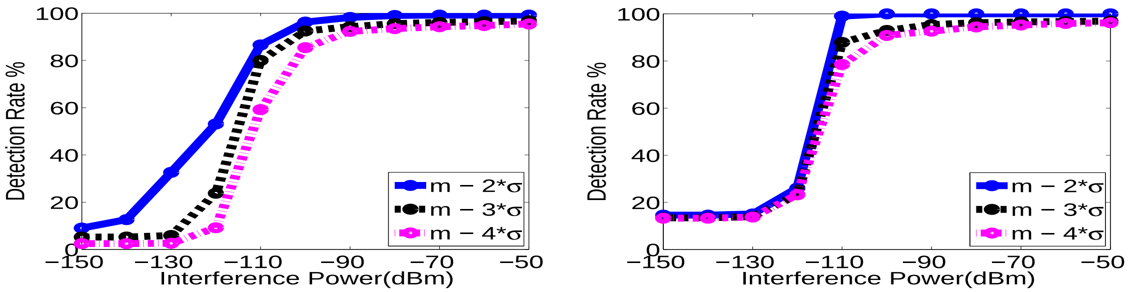
(a) Average C/N_0

(b) Combined C/N_0

Figure 3.8: C/N_0 profile construction: (a) shows average C/N_0 for different trips for a route and (b) shows the constructed C/N_0 profile the route.

of 12 km which includes one underpass and a city road for route 2 with a distance of 6 km. We drove our vehicle in regular traffic at different times of a day for 20 different days over 4 weeks and use the smartphones and the GPS receiver to record the GPS signal for all trips. We split the distance into 20 meters long road intervals which is relatively small and is long enough to get several readings from receivers.

Metrics. In this work, we utilize the mean value of 4 largest SNRs from all satellites, since GPS chips utilize the 4 strongest satellite signals to determine 3-D positions. In the rest of the paper, SNR level corresponds to such a mean value. We consider the case, where the SNR level reported by a receiver is less than the threshold, as a detection when the GPS jamming/interference is present. Based on this, we evaluate the performance of the detection algorithms in terms of following metrics: 1) detection rate (DR) is the percentage of distance intervals in which the detection is successful. For instance, in our route 1 experiment, there are 610 different road intervals (i.e 610 * 20 meters = 12.2 km), and if the detection is successful in x of them, then the detection rate becomes (x * 100 / 610) %. 2) False positive is the case when our algorithm determines there is a suspicious GPS interference activity although it is not coming from a jammer. We note that the detectable interference power is defined as the level where detection rate exceeds 95%.



(a) u-blox GPS receiver

(b) Smartphone

Figure 3.9: Estimated detection rate of our detection algorithm for different interference powers.

Experimental Results

Feasibility Study of Profile Construction. Figure 3.8 shows the average C/N_0 over 4 days (each trip represents one day) versus the relative distances from the starting points of the trips in two cities. We observe that the C/N_0 levels are consistent at same locations across different days. For instance, there is always a big drop in C/N_0 at about 6 km in Figure 3.8(a) which is caused by an underpass during the road trip. We note that the C/N_0 level in smartphones fluctuate more than that in the GPS receiver.

To further illustrate the basic idea, a C/N_o profile could be constructed by combining the C/N_0 values from all trips across different days into each corresponding distance interval. The constructed C/N_0 profile of the testing route is shown in Figure 3.8(b).

Detection Performance. As we discussed in Section 3.3, we can determine whether there are interference signals or not by comparing the newly reported SNR to a fixed threshold. We present the detection rate of our algorithm under different interference power by varying the threshold levels in Figure 3.9. In our profiling algorithm as presented in Section 3.3, the threshold is defined as $\delta = m_{snr} - k * \sigma_{snr}$. We observe the detection rate increases with the increase of received interference power.

k	DIP u-blox stationary	DIP u-blox mobile	DIP phone mobile	False Positive
2	-108 dBm	-103 dBm	-109 dBm	5%
3	-105 dBm	-86 dBm	-91 dBm	1%
4	-101 dBm	-55 dBm	-69 dBm	0.1%

Table 3.1: Detectable Interference Power and False Positive Rate for different thresholds

And the best detection rate is achieved by using k=2. We further present the false positive rate of our algorithm and detectable interference power(DIP) under different thresholds in Table 3.1. We find that low false positive rates are achieved under all cases, indicating the effectiveness of our approach.

The detection range is dependent on signal propagation conditions. We therefore first evaluate detection rate based on the actual received interference power. We further study how to decide on the distance of the jammer. From detectable interference power results in Table 1, we know that if the received power is -109 dBm, with 95 percentile possibility we can detect the jammer. By using the free space path loss(FSPL) model with two propagation exponents 3 and 3.5, we observe the jamming activity could be detected under the distance of 77 and 14 meters respectively.

3.5.3 Roadside Monitoring for Source Vehicle Identification

We next study the accuracy of source vehicle identification through roadside monitoring. To perform a representative case study, in place of an actual jammer, we use a transmitter with similar characteristics in the closest ISM band at 900MHz. This is to minimize the chance of causing any GPS interference. The shift to the 900MHz band primarily means somewhat better material penetration and higher antenna effectiveness. This means that the range in the L1 band will be somewhat shorter, but we believe these 900MHz results are sufficiently close to be useful guidance.

Experimental Setup

Hardware and Signals. As described in section 3.3, we conducted our experiments using USRP N210 at 915MHz band to implement the energy-based interference detector at each monitoring point. The USRPs were connected to laptops streaming data to or from the host processors. A video camera was setup with the USRP at the monitoring point to record videos during the experiments with its time synchronized with the laptop. Specifically, the camera is on an adjacent overpass at each monitoring point, and angled to capture the video of traffics on all lanes within the transmission range of the detector.

Real Road Scenarios. We chose a suburban highway as a representative route, and placed two monitoring points 3 miles apart. The locations were chosen to reflect different lane/road configurations and based on the availability of overpasses for camera placement as well as the availability of safe roadside positions (a parking lot and bus stop). We drove the vehicle carrying the transmitter on the right-most and left-most lanes, and passed by the monitoring point for 20 times, respectively. The experiments were across different days with regular traffic.

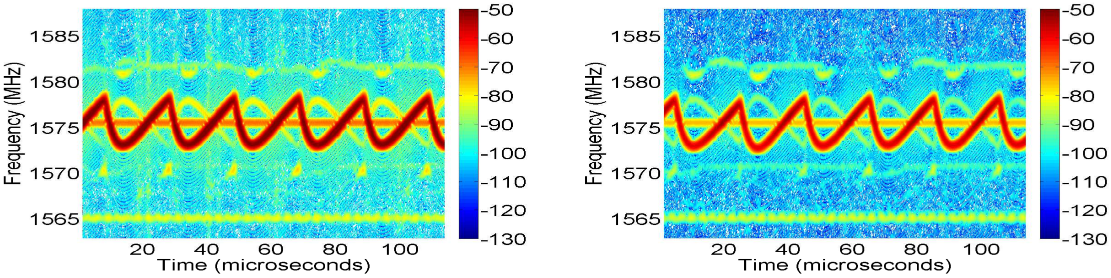
Metrics. We consider the number of candidate vehicles that are in the detection area(s) at the time of detection. If the number is one, and this is our transmitter vehicle, we consider a detection successful. Based on this, we evaluate the performance of our detection and identification approach in terms of the following metrics: 1) detection rate (DR): the percentage of passes during which our transmitting vehicle was correctly identified by our system, and 2) false detection: when a vehicle that likely does not emit interference signals is implicated by our system.

Experimental Results

Validation of the Emulated GPS Jammer To validate this detection approach and ensure a sufficient detection range as well as robustness to high speeds we conducted the following experiment. In addition to the roadside detector we used a second USRP with a Matlab generated (Figure 3.2) interference signal to emulate a jammer. Both were equipped with one omni-directional antenna [30] (824-960 MHz).

Given that the simulated GPS jammer jams at the power of -15 dBm, we were able to detect it 50 meters away, indicting a good chance of detecting GPS jammers from a roadside location. The spectrograms of the received signal at different speeds (Figure 3.10) shows that the chirp shape of the jamming signal remains visible even if the speed is 50 MPH and the received signal strength remains similar at different speeds.

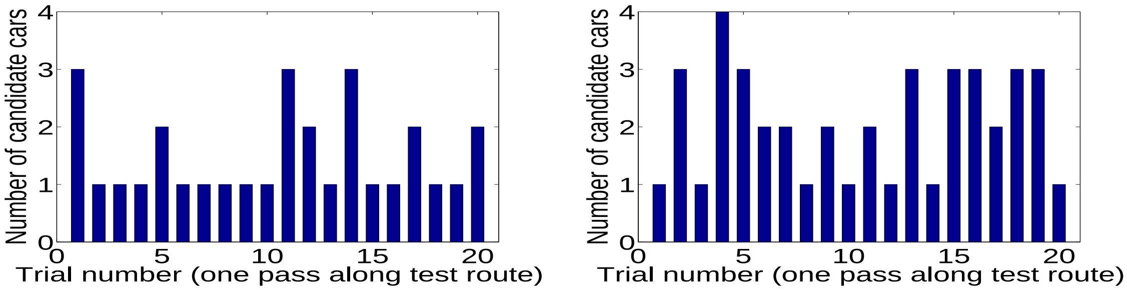
Single Monitoring Point Case. We first consider a single monitoring zone. Provided with the video images from the time of signal detection, we manually identified the number of vehicles inside the detection zone and whether our transmitter vehicle



(a) 0 MPH (static)

(b) 50 MPH

Figure 3.10: Spectrogram of a jamming signal captured by the jammer detector from the simulated jammer transmitting jamming signal at 915 MHz.



(a) Monitoring point 1

(b) Monitoring point 2

Figure 3.11: Number of candidate cars at a monitoring point when jamming activity is detected.

is among them. As shown in Fig. 3.11, at monitoring point 1 there is only a single vehicle in the monitoring zone in 13 out of the 20 passes and 2 to 3 vehicles are present in the remaining 7 trials. Our transmitter vehicle was present during all 20 trials, this indicates our system did not have any false positives and were able to uniquely identify the transmitting vehicle in 65% of the cases. When only considering monitoring zone 2, a single vehicle was present in 7 out of the 20 passes with 2 to 4 vehicles in the remaining ones, yielding no false positives and a lower detection rate of 35%. This is likely due to the larger number of lanes at this location.

Multi-Monitoring Point Case. The detection rate can be significantly improved by combining the information collected from both zones. In particular, in the 16 cases where more than one car was inside the detection zone at either monitoring point 1 or 2, we determined the intersection of the candidate vehicle set from zone 1 and 2, as per the cross validation method when multiple detection zones are present. This resulted in only a single detected vehicle and a correct identification of our transmitter vehicle in all cases.

To investigate these aspects further, we determined the time difference that has accumulated between cars that passed monitoring zone 1 together when they arrived at monitoring zone 2. Figure 3.14 (a) presents a histogram of this time difference. Positive timing means it takes more time for a regular vehicle to travel from monitoring point 1 to 2 than the transmitting vehicle and vice versa. This time difference allows us to judge how large detection zones can become before the detection rate would fall. Our current detection areas could be traversed in about 1-2 seconds at typical speeds. The histogram shows that only a detection zone taking more than 4 seconds to traverse (approximately 100m at 25m/s) would start to yield failed detections.

Detection Zone Size and Detection Rate. The relationship between detection zone size and detection rate is further examined in Figure 3.14 (b). Here the detection time interval represents a detection zone in the time domain (for example, a 50m zone is approximately equivalent to 2s at 25m/s). When this interval is too short, the detection zone may not include the transmitting vehicle and detection rate is low. If it is too long, many other vehicles are also in the detection area, which means unique vehicle identification is not possible and detection rate declines again. In our roadway experiments, detection rate was optimized with a 3-4 second interval, achieving a detection rate of 100%.

Thus, to determine the detection zone size, we drove the transmitting vehicle and passed by each monitoring point 20 times as mentioned in section 3.4. In each pass, when the detector is triggered by the transmitting signal, our system recorded the position of the transmitting car and marked that position in a screenshot obtained from the camera as can be seen in Figure 3.12. These positions let us empirically create a detection zone which covers all of these positions.

Evolution of the Signal Through One Pass. When we were passing by the monitoring point, the detector detects several activities (10 ms long). In Figure 3.13(a), we present how the normalized amplitude of the signal is changing over time for three passes as an illustration. Finally, Figure 3.13(b) presents the received power for each

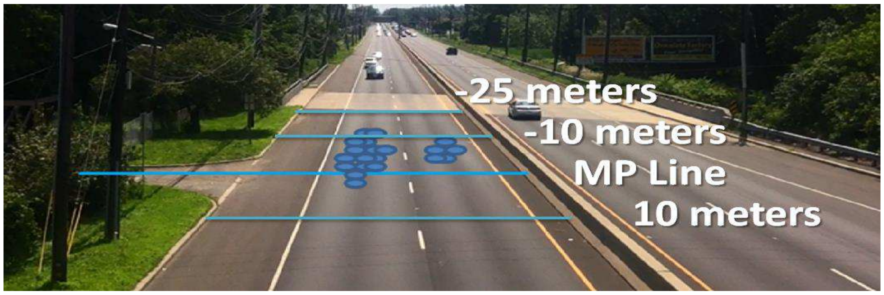
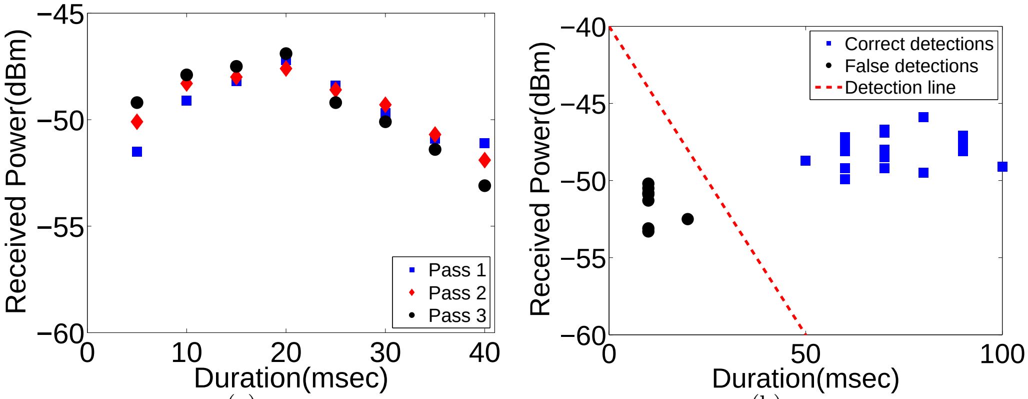


Figure 3.12: Positions of the transmitter vehicle at the time of maximum detection in monitoring point 1.



(a)

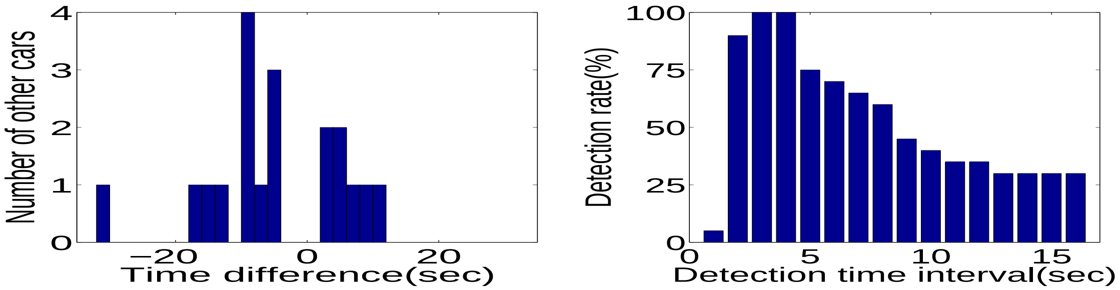
(b)

Figure 3.13: (a) shows the change in received power over time as the transmitting vehicle passing by the monitoring point 1 and (b) shows correct detections and false detections at the same monitoring point.

detected signal with the duration of that signal together with a detection threshold at monitoring point 1. It is clear that in the false detection cases, the durations of the incident and the received power are much less than those in the correct detection cases. Similar observations are obtained from monitoring point 2. Our detection algorithm gives perfect results (precision 1) for both monitoring points.

Discussion

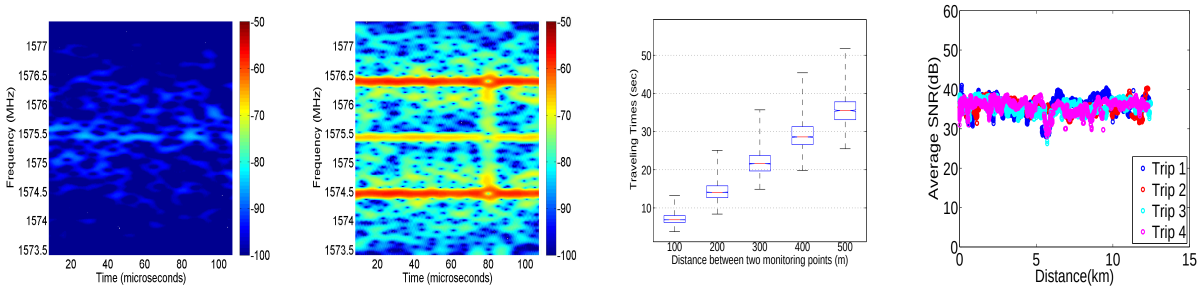
In our experiments, the distance between the monitoring points is about 3 miles and there is no traffic light in between. If the distance between the monitoring points is shorter, then we expect to see more cars to travel together and they would be seen in the monitoring points together. If there are some traffic lights between the monitoring points, it would force the drivers to move in the same groups and the number of common



(a)

(b)

Figure 3.14: (a) Time difference between the jamming vehicle and other suspicious vehicles that are in the detection areas at both monitoring points, (b) The change of detection rate with the detection zone size.



(a)

(b)

(c)

(d)

Figure 3.15: (a) Spectrogram of no jamming activity at location 1, (b) suspicious interference activities detected at location 2, (c) Travel time of vehicles for different interval length, (d) HTC Evo phone for route 1.

cars in both monitoring points is expected to be more than those in our case. Also driving styles can effect the results.

In the evaluation, we have not compared with competing solutions, because we are not aware of an approach that can isolate mobile GPS jammers with the level of resources that were at our disposal (we did not have access to the FAA/FCC teams and contractors described in Newark Airport incident).

Passive Roadside Monitoring

We deployed our detection system in several locations in two US cities for monitoring GPS L1 frequency band (1575.42 MHz) passively on the roadside. We mainly use a interference/jamming detector equipped with a multi-band GPS antennas [55] to continuously monitor the GPS L1 frequency (1575.42 MHz) band. The detector is tuned to log any suspicious signal at the L1 band when its power exceeds the pre-set threshold, which was -60dBm. The suspicious signals are studied through offline analysis.

To perform extensive roadside monitoring, we want to identify the interfering GPS signals on public roadways. With this purpose, we have selected several locations on three kinds of roads: a major highway, one of the busiest toll road, and an urban road for passive monitoring of GPS frequency band. In total we monitored the roadways passively for approximately 200 hours in the 5 locations in NJ and South Carolina.

In the 200 hours monitoring at five monitoring points, we detected two suspicious interference incident at the location 1 (on a busy toll road) and the location 2 (on an interstate highway) respectively. As shown in Figure 3.15, there was high energy at the L1 frequency band in both of the two incidents.

We also emphasize that although, it is not clear if the signals detected in passive monitoring are coming from a real jammer, these results show that significant interference events happen in the L1 frequency band and even non-jamming interference can cause problems.

3.6 Discussion

Our goal is to design a low-cost detection mechanism that can be mass deployed in roadways, and thus our jamming detection mechanisms are designed to identify existing off-the-shelf GPS jammers that continuously emit interference. Future jammers that may emit the interference for a short period of time could still be caught by the mobile detectors (e.g smart phones) since they could still be reporting any outages or SNR anomalies even the strength of the jamming signals change over time. With such reports, the path of the jammer car can be interpolated to cameras for identification. The probability that jammer would be detected by a stationary monitoring point on a single pass is smaller but after a sufficient number of passes (say, for a commuter), detection would still be likely.

Opportunistic Crowdsourcing. While not every individual user will be motivated to install our mobile detector app for jammer detection, we envision that this mechanism can still be deployed in police vehicles, public transit systems, delivery truck fleets, etc. It could also become part of future connected vehicle standards or be integrated in smartphone platforms with careful energy management.

GPS Spoofing. Apart from GPS jamming, another security issue is `GPS spoofing` attacks, whereby attackers transmit fake GPS signals to fool the GPS receivers[58]. Note that GPS spoofing is outside the scope of this paper and we focus on designing a practical solution for detecting GPS jammers. Nevertheless, our method is complementary to GPS spoofing detection strategies.

3.7 Conclusion

Several incidents caused by in-vehicle jamming devices have illustrated their serious impact on the availability of the navigation services in critical infrastructure. To monitor and reduce the use of in-vehicle GPS jammers, we presented a low-cost jammer identification system that can be mass deployed in roadways to automatically detect and identify the vehicles with GPS jammers. Instead of covering all types of possible jammers, our jamming detection mechanisms focuses on achieving wide geographical coverage against the most prevalent off-the-shelf GPS jammers, which continuously emit interference. The dedicated detectors, however, could be extended for more sophisticated jamming signals. The key components of the system are monitoring stations (which are equipped with directional antennas and cameras) and mobile detectors (e.g., smartphones). Using an off-the-shelf software-defined radio (USRP) to emulate GPS jamming signals, we conducted a case study evaluation of our system with multiple trial drives on local highways in 2 US cities and found the monitoring stations effective.

We also demonstrated that by constructing location-based profiles of expected signals, our mobile detector component can detect interfering signals based on measurements that are readily available in most GPS receivers, including the ones in smartphones.

Thus, it is possible to detect jammers via crowdsourcing. While not every individual user will be motivated to install our mobile detector app, we envision that this mechanism can be deployed in law enforcement vehicles, public transit systems, etc. End users may also volunteer to report data while using GPS navigation services, and thus provide opportunistic crowdsourced data. The positive results that are obtained when validating the most important components of our solution, i.e., the Energy-based monitoring point and the SNR-based mobile detector, are promising. They indicate the validity of a full-fledged system that consists of a centralized vehicle identification component integrated with monitoring points and mobile detectors.

Algorithm 1 Mobile Detector

```
function PROFILESNR($(n, d)$)
                                        ▷ record time series data for n satellites
    $< time, SNR_{sat}, lat, lon >= record\_data(n)$
    $p = calculate\_path\_distance(lat, lon)$
    for each path segment $p_i$ with length d do
        $sat_{snr}(p_i) = get\_sat\_snr(SNR_{sat})$
        $m_{snr}(p_i) = average(sat_{snr}(p_i))$
        $\tau_{snr}(p_i) = mode(sat_{snr}(p_i))$
        $\sigma_{snr}(p_i) = std\_dev(sat_{snr}(p_i))$
        $save\_profile(p_i, m_{snr}, \tau_{snr}, \sigma_{snr})$
    end for
end function
function DETECTSNRANOMALY($(n, k, p_i)$)
    $sat_{snr} = read\_sat\_snr(n)$
    $m_{snr} = average(sat_{snr})$
    $\tau_{snr} = mode(sat_{snr})$
    $\sigma_{snr} = std\_dev(sat_{snr})$
    $< m_{pr}, \tau_{pr}, \sigma_{pr} >= get\_profile(p_i)$
    $\delta = m_{pr} - k * \sigma_{pr}$
    if $m_{snr} + (\tau_{pr} - \tau_{snr}) \leq \delta$ then
        $jamming\_status = verify\_all\_sat()$
        if $jamming\_status = true$ then
            $[lat, lon] = get\_location()$
            $send\_report(lat, lon, m_{snr})$
        end if
    end if
end function
```

Chapter 4

PRE-TRIP DRIVER IDENTIFICATION FROM IN-VEHICLE DATA

4.1 Overview of Chapter

As vehicles are becoming programmable and connected, we expect that an increasing number of applications will enjoy access to internal vehicle data. Modern automobiles contain hundreds of sensors and actuators that exchange data on internal buses. A small part of this data has already been exposed in the OBD-II standard but the majority has to date been used only internally. Recently, car makers have been experimenting with opening more of this information to smartphone or in-car apps [5, 6]. Such data is also increasingly accessible through telematics services and could potentially be processed in the cloud.

Driver specificity of data. One relevant question in this context is how driver-specific the data is. How easily can different drivers of a vehicle be distinguished from such in-vehicle data—or, more precisely, what is the minimal amount of data necessary to effectively distinguish drivers? The answer to this question will help in understanding the feasibility of building driver-specific applications for the many vehicles that are used by multiple drivers. We focus on shared vehicles in a professional or commercial setting, and personal use setting, where applications include personalization of vehicle settings (e.g., automatically adjusting entertainment, HVAC, transmission, or suspension configurations to driver preferences). automated vehicle use logs, driver-dependent pay-as-you-drive insurance, or unauthorized vehicle use detection (where a vehicle might notify owners when it encounters an unexpected driver). The answer to this question will also contribute to an understanding of the privacy implications of such in-vehicle data. The more easily drivers can be distinguished, the less anonymous one can expect this data to be.

Existing work. Existing product solutions to distinguish drivers usually require a token, such as a smart key fob, that the driver carries. Such systems are robust only if every driver consistently uses a separate key fob. They are often limited to two keys due to cost and cannot distinguish drivers when keys are shared in commercial settings with more than two drivers. The academic literature has also explored how mobile devices in vehicles can determine whether they are used in the driver area of the vehicle, which would usually indicates that their owner is the driver [79, 77, 46]. Such techniques can also identify the driver, but depend on specific interfaces to the vehicle, keeping the phone close-by while driving, or the usage of more advanced wearable devices.

Existing work towards understanding the driver-specificity of vehicle data has been limited to a few parameters such as vehicle movement and steering inputs [28]. More work exists, of course, on location data which can also be obtained from cell phones inside a vehicle [38, 39, 37]. Such techniques require a complete trace of vehicle data from a longer trip. These results therefore tell us that drivers can be identified in longer sets of data but do not identify a minimal set of data for identification or convey a good sense about the ease of this identification.

A pre-trip profiling approach. In this paper, we address these questions by examining in-vehicle data streams and exploring a driver differentiation system that can rely on minimal time sequences of in-vehicle data. We define minimal time sequences in terms of the amount of time that has passed since approaching the vehicle for a new trip. This allows us to understand the driver-specificity of different types of in-vehicle data generated over the course of a trip and it is also consistent with the personalization use case, wherein the vehicle rapidly needs identify the driver to switch to the driver's preferences.

We first examine sensor information shared on the vehicle bus for its driver specificity. This analysis includes data that was so far largely unavailable to external entities. We find that, in addition to the expected driving telemetry data generated while the vehicle is steered, the data streams contain a rich set of fields that reflect other driver actions, such as fastening seat belts, closing doors, or changing HVAC settings. A particularly revealing burst of this data occurs at the start of a trip, before the vehicle starts moving. Based on this insight, we explore a pre-trip data profiling approach and compare it with the use of more conventional driving telemetry data. Pre-trip data are due to driver actions taken in the first 20 seconds after entering the vehicle and include: the time at which the vehicle door was closed, the vehicle was started, the seatbelt was fastened, and the brake pedal was released. Driving telemetry data includes vehicle speed, acceleration/deceleration patterns, braking patterns at stop signs, or turn signal use. We consider a system that monitors these events on the vehicle bus, extract timing features, and distinguish different drivers of one vehicle using a classifier. We conduct controlled experiments with 24 volunteer drivers, who take a test vehicle along a pre-defined campus course. We also collect and analyze a real-world dataset of 480 trips from five shared university mail-vans spread over 12 weeks. Finally, we conduct a three-week study with households and present the results. The experiments reveal that the data when starting the car is actually more revealing, than the driving behavior on the roadway.

In summary, the salient contributions of this work are the following:

- accessing a rich set of in-vehicle sensor data through a custom CAN bus interface and examining its driver-specificity; this explicitly includes data that was so far inaccessible through interfaces such as OBD-II and OpenXC.

- designing classifier features and a system that allows distinguishing drivers based on a minimal set of in-vehicle sensor data, with no additional hardware cost.

- evaluating the system with data from 480 real-world trips collected over 3 weeks from five university mail vans, with 24 drivers in a controlled experiment, and 103 trips with four drivers across two households.

- finding that data from the vehicle start is particularly specific to individual drivers, allowing our system to achieve 91% of accuracy within 20s after the driver enters the vehicle in the real-world mail van experiment.

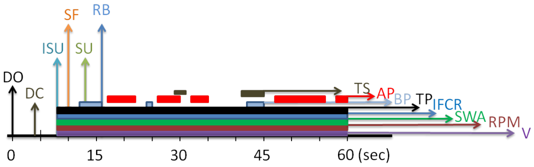


Figure 4.1: Timing events of the first minute of a trip.

4.2 Driver Differentiation

We seek to identify a minimal time-sequence of in-vehicle data to distinguish drivers by identifying events that are closely tied to driver habits and behaviors but minimally affected by driving conditions and other traffic participants.

4.2.1 Selection of Pre-Trip Events

We analyzed the available in-vehicle data and identified 14 fields that are available in many vehicle models and whose value commonly changes early in a trip. These fields are illustrated in the timeline in Fig. 4.1, derived from one example trip with a mail van. Note that the vehicle generates data even before it starts moving and that many of these initial events directly correspond to driver actions such as *door opening (DO)*, *door closing (DC)*, *starting the ignition (ISU)*, *seatbelt fastening (SF)*, *shifting gear (SU)* and *releasing the brake pedal (RB)*. As the engine is turned on and the vehicle begins to move, additional driving and telemetry data streams indicating steering wheel angle (SWA), engine revolutions per minute (RPM), vehicle speed (V), and acceleration (AP) values become available.

We hypothesize that the pre-trip events generated before the vehicle moves are not only available early after entering the vehicle but are also particularly distinctive because they are largely dependent on habit and unaffected by the road configuration and actions of other traffic participants. While the type of events does not differ across drivers, the order and precise timing of these actions is mostly determined by habit. To what degree one turns the steering wheel while leaving a parking lot is often affected by the presence of other obstacles and vehicles at that particular location. The relative

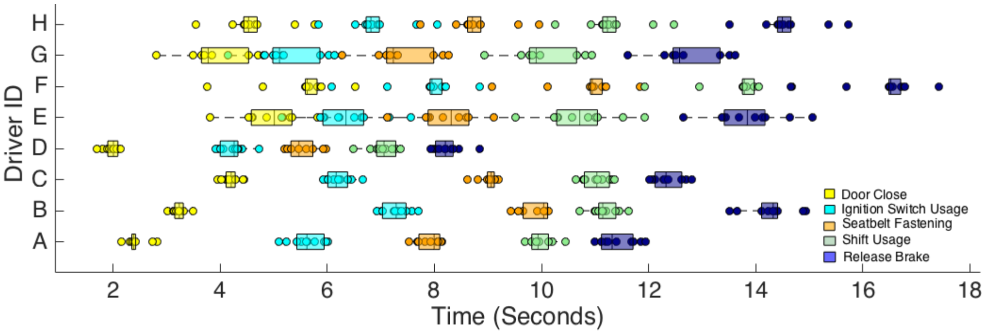


Figure 4.2: Pre-trip fields timeline - Controlled Experiment.

timing of ignition start and seatbelt fastening, in comparison should not depend significantly on these external factors. These steps, their specific order (sequence) and their timing interval should therefore, be helpful in creating a minimal driver profile.

To support the hypothesis, we conducted a preliminary experiment with eight drivers. The drivers were instructed to drive a Cadillac CTS and complete a loop in the parking lot. Each driver repeated the experiment 10 times. For this preliminary experiment, drivers were asked to consistently follow their regular habits, to reveal possible distinct patterns across drivers. We describe our in-the-wild experiments in the evaluation section. Figure 4.2 shows the pre-trip event timing collected from those drivers in a scatter plot with quartiles marked for each event type. Time zero is defined as the door open event, the first event related to this trip. The data shows that the relative timing is quite distinct across drivers, even when drivers started the vehicle in the same controlled test situation. We also observe that these pre-trip events occur within 20 seconds after opening the door in all cases.

4.2.2 System Overview

Based on the aforementioned insights, we consider a driver differentiation system that seeks to distinguish drivers using a minimal time sequence of in-vehicle data. The system primarily consists of three components, vehicle bus data capture, feature extraction, outlier rejection, and driver differentiation. It obtains in-vehicle data, particularly pre-trip events, through a CAN bus interface. The feature extraction module scans this data to identify the start of a new trip, extract event timings, and construct a feature vector. This vector is then examined for outliers before being processed by a classification algorithm, that matches the feature vector to profiles constructed from past trips of the same vehicle. In applications where labeled training data does not exist, the use of unsupervised classification techniques is also possible, and presented in Section 4.4.

There are multiple possible realizations of such a system in practice and we illustrate the design space through the following examples. Potentially, all the above mentioned components could be directly embedded in the vehicles, perhaps as part of a driver personalization feature that is transparent to users. Such built-in components could directly access the CAN bus and acquire the necessary data from there. A second possibility is that the feature extraction and driver differentiation functions are executed in cloud-based car maker applications and receive access to the vehicle data stream over increasingly available wireless broadband data connections to vehicles. For commercial settings, where the same driver may end up driving different vehicles on different days, running the driver differentiation module on a remote server is more suitable. However, for drivers who share the same car, such as members of a family, the computation can be done locally on the vehicle itself. A third possibility is that the components are realized within a third-party application that acquires sensor data through a vehicle manufacturer developer API. Depending on the availability of suitable APIs and other considerations, such applications could reside either on an app platform in the vehicle or in the cloud. It is also possible that some components are located on a mobile device brought into the vehicle, which pairs with a vehicle interface that provides access to the vehicle bus. In our experiments, we focused on this last option, a smartphone interface that allows data capture, with feature extraction and classification performed in the cloud.

4.2.3 Driver Profiling

We create a robust driver profiling approach, that extracts features resilient to precise driving style but derived from driving habits.

Feature Selection

Our features emphasize pre-trip data that represent seemingly innocuous habitual patterns of every driver, which are crucial in distinguishing them from others. For every pre-trip event k, the corresponding feature is defined as Δt_k, which is the time difference between the occurrence of event k and $k-1$. Specifically, our pre-trip feature vector f_{pt}, is defined as follows.

$$f_{pt} = [\Delta t_{DC},\ \ \Delta t_{ISU},\ \ \Delta t_{SU},\ \ \Delta t_{SF},\ \ \Delta t_{RB}]$$

Here, Δt_{DC} represents the time difference between the occurrence of the (driver) Door Close (DC) event and the occurrence of a reference starting event, which marks the beginning of the vehicle data stream for the new trip. Unless otherwise mentioned, we use the (driver) Door Open (DO) event as the reference event, since this was the first observable event on the vehicles that we experimented with. Similarly, Δt_{ISU}, Δt_{SU}, Δt_{SF}, and Δt_{RB} represent the time difference of the Ignition Switch Usage (ISU), Shift Usage (SU), Seatbelt Fastened (SF), and Release of the Brake pedal (RB) events to events DC, ISU, SU, and SF, respectively. The feature vector is always constructed in the same order irrespective of the actual order of events. In considering time intervals between specific events, we capture their relative occurrences. The DO and DC events are usually triggered when the driver enters the car. The ISU event represents starting the engine. The Seatbelt Fastened event occurs when the driver fastens their seatbelt. Shift Usage refers to changing the setting on an automatic transmission from park mode to another mode, often drive or reverse. Lastly, the Release Brake event marks the instance when the brake pedal was released to start driving.

In addition to pre-trip fields, we also extract features from the driving fields for comparison. Our driving feature matrix f_d, is defined as follows:

$$f_d = \begin{bmatrix} bp_1 & ap_1 & rpm_1 & tp_1 & ts_1 & v_1 & swa_1 \\ bp_2 & ap_2 & rpm_2 & tp_2 & ts_2 & v_2 & swa_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ bp_N & ap_N & rpm_N & tp_N & ts_N & v_N & swa_N \end{bmatrix}$$

Each row in f_d is a feature vector at some time i. Every feature is the value of the sensor as read from the vehicle bus. bp_i and ap_i represent how far the Brake Pedal and Accelerator Pedal are pressed at time i. Similarly, rpm_i denotes the engine Revolutions Per Minute, and tp_i represents the Throttle Position. Turn Signal is signified by ts_i. v_i and swa_i stand for the values obtained from the vehicle Velocity and Steering Wheel Angle sensors. The frequency and range of these sensors is shown in Table 4.1.

Adaptive Outlier Rejection

The normalized feature vector is used as an input to a learning algorithm. Using all the features in f_{pt} can help us differentiate drivers, but is easily affected by slight variations in event timings. To account for day to day driving behavior, we devise an adaptive outlier rejection technique. We note that although pre-trip sequences are peculiar to each driver, they are not always exactly comparable. Circumstances may arise when drivers break out of their pre-trip routine, causing a larger than usual delay in one of the events, as a result of which the subsequent events might be delayed as well. A common example is when drivers are interrupted by a phone call. Consequently, the corresponding feature is an outlier compared to past values of that feature for the same driver. In such cases, we do not want the outlying time interval to affect our classification. The adaptive outlier rejection technique is designed to identify such outliers at test time, and adapt to them. We examine our pre-trip feature vector for outliers by comparing each feature value to the distribution of that feature. We deem a particular feature as an outlier if it lies one standard deviation or more above the mean for that feature. In this case, we dispose of that feature, but retain the subsequent time intervals (features). Note that we only remove at most one feature from f_{pt}. In case of multiple outliers, we discard the feature that is furthest. The feature vector is now reduced to *n-1* values.

Two-step Driver Validation

For driver validation at run time, we perform outlier rejection and model selection. We use supervised learning for differentiating between drivers. During the training

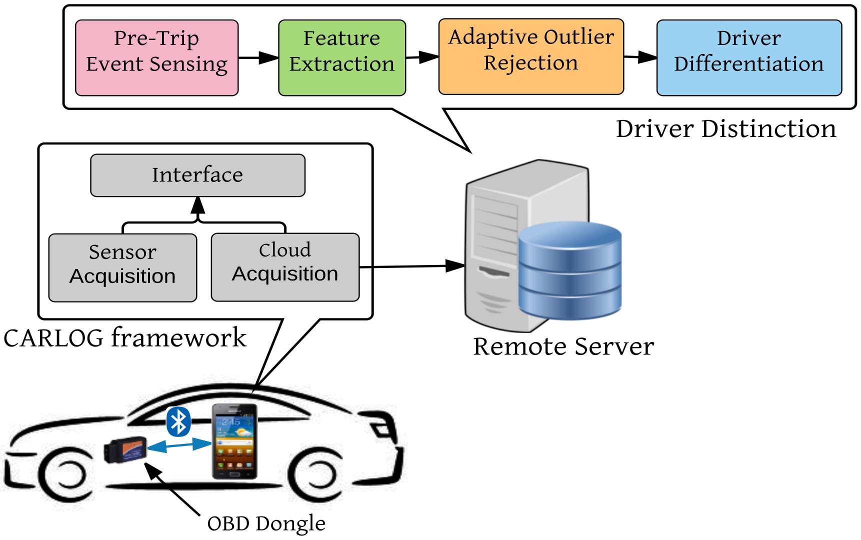


Figure 4.3: Our system implementation.

phase, we train a classifier using all the features (n=5), called the complete model. In addition, we fit separate models on different combinations of all but one feature in the training data. We achieve this by removing one column (feature) at a time. We refer to these models as partial learners. Partial learners learn from $n-1$ features. Note that we train partial learners on the entire test set and not just those with outliers. Removing at most one feature at a time limits the total number of models to $n+1$. If an incoming pre-trip sequence has no outliers, i.e. it conforms to within one standard deviation around the mean of the feature, we use all the features in f_{pt} and test it against the complete model. When we observe an outlier in incoming trip data, say a long delay for event k, we dispose of the Δt_k, but retain the subsequent time intervals. Our algorithm classifies the driver by testing this feature vector (with n-1 features), against the corresponding partial learner.

We use Support Vector Machine (SVM) with a linear function as our learning algorithm, with 5-fold cross validation. We observed this simple learner to give the best performances for our data as compared to other kernels, such as cubic and gaussian, and different learning algorithms like k-means clustering and decision trees. Additionally, this simple approach is computationally lightweight and suitable for real-time driver differentiation on COTS mobile devices. In an automated vehicle use logging, the set of

Pre-trip Fields	Frequency (Hz)	Range	Driving Fields	Frequency (Hz)	Range
Door status (DO & DC)	10	Boolean	Brake pedal (BP)	10	0-100
Ignition switch status (ISU)	10	Boolean	Accelerator pedal (AP)	50	0-100
Seatbelt status (SF)	10	Boolean	Revolutions per minute (RPM)	10	0-16000
Shifter position (SU)	40	Integer(1-6,13,14,15)	Throttle position (TP)	10	0-100
Parking brake active	100	Boolean	Turn signals (TS)	Event	Boolean
			Vehicle velocity (V)	10	0-255 kmh
			Steering wheel angle (SWA)	100	0-1340°

Table 4.1: Fields captured from the CAN bus.

Pre-trip fields	Mid-size sedan		Luxury vehicle		Van	
	Before ISU	After ISU	Before ISU	After ISU	Before ISU	After ISU
Door status (DO & DC)	χ	✓	✓	✓	χ	✓
Seatbelt status (SF)	χ	✓	χ	✓	χ	✓
Shift status (SU)	χ	✓	χ	✓	χ	✓
Release break (RB)	✓	✓	✓	✓	✓	✓

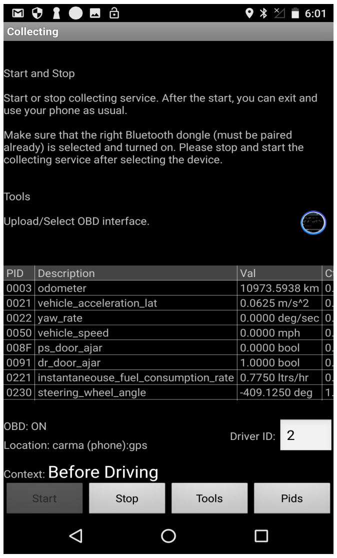
Table 4.2: Availability of pre-trip fields for different vehicle models.

drivers is typically known, and labeled training data may be available from earlier manual logs. In other applications, such as vehicle personalization, the number of drivers is far less. To collect training samples, the data logging application prompts the driver to mark the ground truth with a simple screen touch at the end of each trip.

4.3 Implementation

We have implemented the entire system using a custom OBD-II scan tool (dongle), a smartphone and a remote server. We place a smartphone in the vehicle that can communicate with the dongle over Bluetooth, as shown in Figure 4.3. Data can be requested through the dongle, by using Parameter IDs (PIDs). The dongle has been specifically designed for research purposes to make available a large set of internal vehicle bus fields that are not yet available through OBD-II or other interfaces such as OpenXC. The dongle sends a PID over the vehicle bus to request data.

We use the CARLOG [44] framework on the smartphone, which is a programming framework, for accessing sensor data from the vehicle. It houses a query optimizer that eases the task of querying, capturing and parsing low level sensor information



Start and Stop

Start or stop collecting service. After the start, you can exit and use your phone as usual.

Make sure that the right Bluetooth dongle (must be paired already) is selected and turned on. Please stop and start the collecting service after selecting the device.

Tools

Upload/Select OBD interface.

PID	Description	Val	C
0003	odometer	10973.5938 km	0.
0021	vehicle_acceleration_lat	0.0625 m/s^2	0.
0022	yaw_rate	0.0000 deg/sec	0.
0050	vehicle_speed	0.0000 mph	0.
008F	ps_door_ajar	0.0000 bool	0.
0091	dr_door_ajar	1.0000 bool	0.
0221	instantaneouse_fuel_consumption_rate	0.7750 ltrs/hr	0.
0230	steering_wheel_angle	-409.1250 deg	1.

OBD: ON

Location: carma (phone):gps

Driver ID: 2

Context: Before Driving

Figure 4.4: Logging Application for In-vehicle sensors.

from vehicles, and provides an interface for applications to access this information. We use a battery conscious smartphone application, shown in Fig 4.4 to record the sensor readings in our testbed of vehicles. The app continues logging until the connection to the vehicle is lost or the app receives no new sensor readings within 1 minute. In this timed-out state, the app closes the connection with the vehicle data port for 30 seconds; after this period, it reconnects to the vehicle, which restarts the cycle. The app sends specific predefined PIDs to the OBD dongle, which then requests it to the CAN bus.

The application initializes a subscription request to receive data from the sensors required for our pre-trip sensing. Among other things, it lets us define the sensors and their associated frequency, in samples per second. It captures data received from the dongle over extended period of time, handles all the incoming data, parses it and records it to local storage. The timestamp for this data is applied in the application when the event reaches the Carlog framework. It does not always accurately reflect the event time, but we expect the error to be small compared to event time differences that we use for our algorithm. In addition, the phone is connected to the internet and updates all this data to a remote server, via WiFi or cellular service. This ensures simultaneous sensing and uploading of the required data fields. For our proposed scheme, we access the on-board sensors on vehicles. Table 4.1 lists some of the sensors whose values are acquired by a smartphone with a bluetooth dongle, through the On-Board Diagnostics

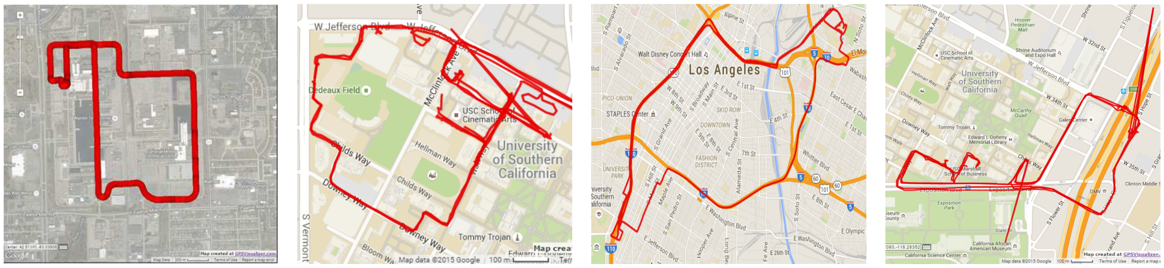


Figure 4.5: (a) Controlled test environment, (b) Mailvan first test environment, (c) Mailvan second test environment, (d) Mailvan third test environment

(OBD-II) port. We also implemented a real-time driver differentiation application for Android smartphones. The classification model is trained offline and loaded on to the smartphone for real-time classification as the driver approaches the car and following pre-trip events.

4.4 Performance Evaluation

For evaluating our system, we aim to answer the following questions:

- What is the the minimal amount of data required to accurately identify a driver?
- How is accuracy affected with increasing number of drivers?
- Which vehicle data fields are most useful for prompt and accurate driver differentiation?
- How does training size impact system performance?
- How does unsupervised driver classification compare to supervised classification?
- How does driver differentiation adapt to drivers within the same household?

To gain an understanding of driver behavior, we carried out the following experiment.

4.4.1 Experimental Setup

Hardware and Signals. We used a 2008 Cadillac CTS vehicle (test vehicle), an LG Nexus 5 phone and a custom OBD scanner (dongle) to extract data from the vehicle. This custom dongle provides access to a richer set of vehicle data streams, compared to standard dongles. During the experiments, the smartphone is located inside the vehicle and the dongle is plugged into the OBD-II port. We conducted an IRB approved study, and used coded data for drivers instead of their actual identities. All participants are 18 years old or older and have a valid driver's license in United States.

Metrics. In our work, we evaluate the performance of driver detection algorithms in terms of accuracy and data length. Accuracy is the ratio of correctly identified number of drivers to the total number of drivers. Data length represents how much portion of the data was used for the identification process.

Controlled Experiment

To evaluate the performance of our system, we first use a dataset that was collected by 16 volunteers who drove the test vehicle on a 3 mile long road. The drivers were requested to drive as they normally would, and were not provided any instructions. They were allowed to make any adjustments they wanted, for example mirrors and seats. The only instruction provided to the drivers was to drive as usual along the given route. Most volunteers were not aware of the vehicle data that we were collecting other than it being related to driver's behavior. During this experiment, we monitored the activity of about 20 sensors, that communicate different types of vehicle state information. These sensors describe the state of electronic vehicular subsystems such as engine start events, cabin climate control and trip events such as speeding, braking, throttle positions and many others. Each driver is asked to drive a pre-defined path 10 times during the same day to create the database that consists of 160 traces. The test path for controlled experiment is shown in Figure 4.5 (a).

Mailvan Experiment

In the mail van driver differentiation tests, we compare the results of our classifier to anonymized versions of the USC Mailing Services Department driver records. These documents contain mandatory vehicle access logs that specify the times when drivers acquire and release a van. Drivers sign-out a single van before leaving the warehouse, and sign-in that van only after returning to the warehouse. During a day, between 1 and 5 drivers will use a single van. Each day, the drivers initially sign-in vans between 6 am and 8 am, and sign them out sometime around 5 pm. The vans are utilized for most of the working business hours, with delays between driver changes ranging from instantaneous to 60 minutes.

The USC Mail Service employees use the vans to traverse specific routes for delivering incoming and outgoing university postage. Each van is associated with a distinct set of routes, which are typically related by location. The three most common routes are shown in Figure 4.5 (b)-(d). The drivers use one van for example, to travel North East from the warehouse towards the USPS office in the morning and a nearby satellite campus in the afternoon. While traversing these routes the driver may make several stops to service multiple buildings on the campuses. These stops typically last between 10 and 60 minutes, and require the drivers to turn off and exit the vehicle. We consider each segment a different trip if the vehicle was turned off at the end of a segment. The mailvan dataset is the result of measuring the electronic subsystems of 7 USC Mailing Service fleet vehicles, over a period of 6 months. Out of the 6 months of vehicle data, we obtained 3 weeks of mail vehicle access logs for each USC Mail Service employee. Digitizing and anonymizing these driver logs requires manual effort, which limited our access to only 15 days worth of records. In an effort to maximize the number of driver changes included in our study, we selected the 3 weeks of measurements from our dataset having the most number of active vans and total number of sensor readings. These 3 weeks worth of data detail the actions of at least 5 vans each day, during the months of September, October, and November. This dataset includes 480 trip start trails that we are mainly using in our algorithm.

4.4.2 Driver Differentiation Evaluation

We evaluate the accuracy of the proposed driver differentiation algorithm with respect to duration since entering the vehicle, various classification techniques, training size, dataset size and importance of each pre-trip field. Unless otherwise specified, for supervised learning we use the 2-step driver validation with a 5-fold cross validation.

Identifying minimal data

With longer driving traces a driver can almost certainly be identified by modeling the vehicle speed, analyzing the destinations visited and the routes taken. This analysis, however, could take several minutes, or may be even hours. Prompt identification of drivers is pivotal for many personalization applications. We compare pre-trip fields and driving fields. Pre-trip fields are a result of fixed actions any driver undertakes before starting a drive. It is a finite set, the duration of which lasts only about 20 seconds or less. Driving fields on the other hand, could be collected for minutes or hours to accurately recognize a driver. We wish to explore the smallest time series of events that can provide reasonable driver differentiation accuracy. From Figure 4.6, it is evident that using only the pre-trip events, we can differentiate drivers with a higher than 90% accuracy, under 20 seconds. Note the zoomed in version that plots the accuracy achieved by using the pre-trip events with respect to data length in seconds. This differentiation occurs before the driver actually starts driving, and well in time to support personalization applications and modifying vehicle settings based on preferences. Note that a baseline approach using only driving data requires approximately 10 min to reach the same accuracy that can be obtained from pre-trip data in 20s. On the other hand, driving data is useful to further improve the accuracy. Processing driving data in addition to the pre-trip data drives the accuracy to 98% within 10 mins. This strengthens our belief that pre-trip events are significant in determining driver behavior, and can be used effectively to perform such behavior based distinction in one-tenth the time proposed by prevalent driver recognition techniques. *The minimal duration of data required for* 90% *driver distinction accuracy, would thus be less than 20 seconds.*

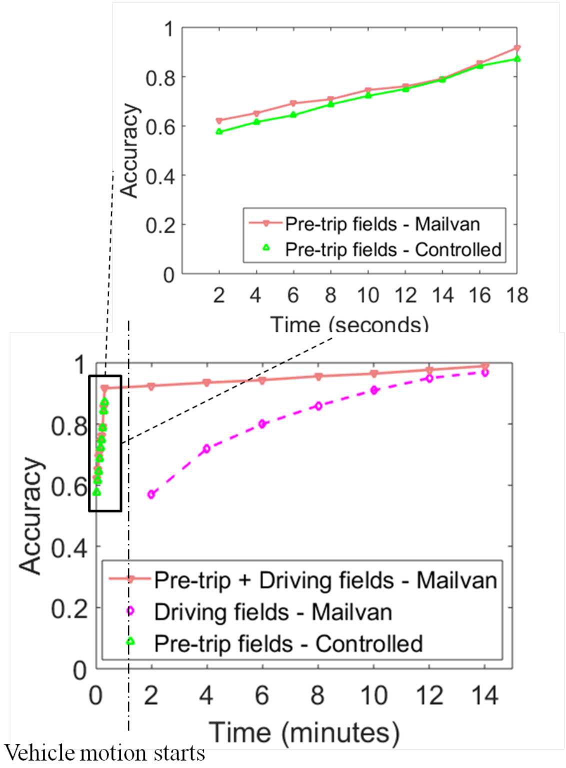


Figure 4.6: Accuracy over trip length. Using only pre-trip fields our system can achieve 90% accuracy in less than 20 secs.

Effect of increasing number of drivers

We take a step further in understanding these pre-trip fields, and their sequence. We asked 16 drivers to carry out the start-up process 10 times, and to drive on 3 mile route. This route was the same for all drivers, and the vehicle was in the same situation at the start of the drive. Since all the drivers were driving on the same route under almost similar traffic situations, and driving the same test vehicle, it might seem hard to accurately differentiate drivers with driving data alone.

It is noticeable from Figure 4.7, that most drivers do not have a clearly separable pattern for pre-trip fields and the time spacing between them. Even for the same driver, these events possess significant variance from the time the driver enters the car. With a large number of drivers, distinction becomes challenging due to high variance for each driver and apparent similarities in the pre-trip event timelines. This is the dataset we used to measure the performance of our algorithm. This emphasizes that

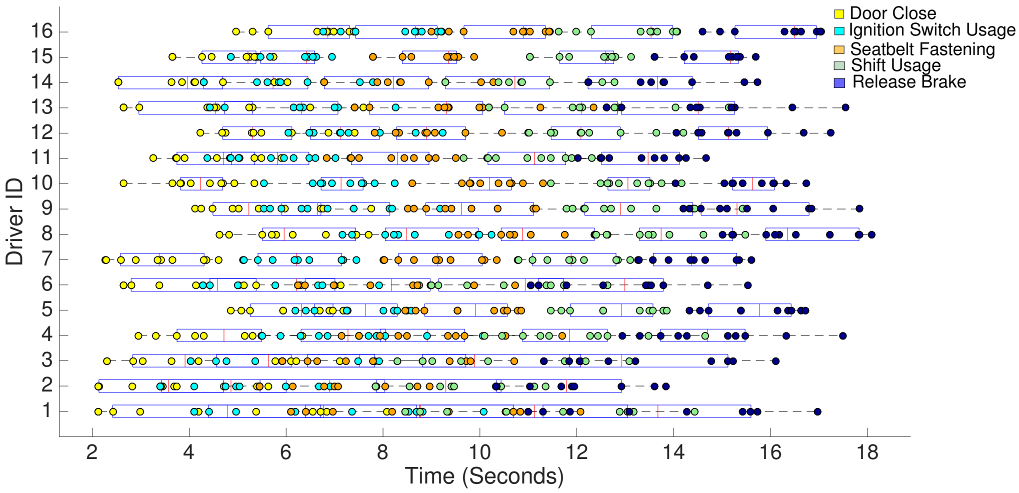


Figure 4.7: Timeline for pre-trip fields.

even with minimalistic pre-trip fields, driver distinction cannot be performed by basic thresholding, and justifies the need for a learning component for understanding driver specificity. For a multi-class classification problem, such as ours, a confusion matrix is commonly employed to demonstrate the classifier performance. Figure 4.8(a) shows the confusion matrix for the mailvan dataset, and Figure 4.8(b) shows the confusion matrix for the controlled set. It can be seen from the figure that our classifier attains high accuracy levels, and does not misclassify drivers.

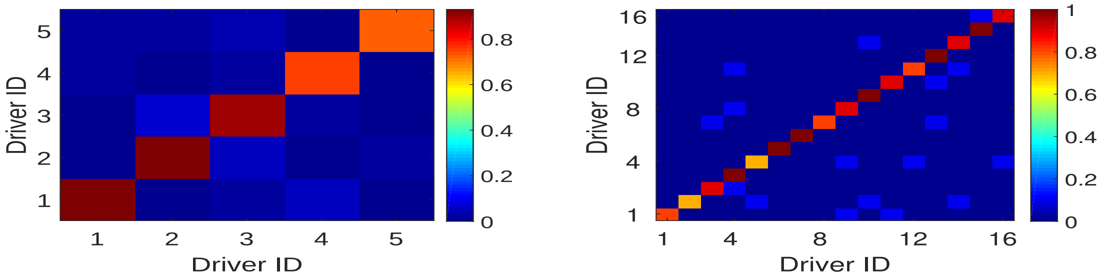
Next, we seek to observe the effect of number of drivers in the dataset, on the classifier performance. In the mailvan dataset, we have traces for five drivers and in the controlled experiment, we have traces for 16 drivers. Figures 4.9 (a) and 4.9 (b) show how accuracy improves when we focus on only a subset of drivers, and it reduces slightly as the number of drivers increases. We observe that, we can get up to 96% accuracy when differentiating four drivers in controlled experiment, and 98% when differentiating two drivers in the mailvan experiment. Our system accuracy declines by about 4% when the number of total drivers is increased from 4 to 16 in the controlled experiment, and by about 7% when the number of drivers increases from two to 4, in a real-world mailvan experiment. *Based on these results, we infer that system accuracy declines as the number of unique drivers in the dataset increases.*

Analyzing individual field importance

The next question we want to investigate is which data fields are most influential in rapidly differentiating drivers. In doing so, we first focus on the Door Open event as the origin event because it is the first event to occur, i.e at time=0. We compute the time difference between each individual event and the origin event. In addition to the aforementioned pre-trip fields, we add another field ACC, which represents the time taken by the driver to go from a speed of 0 mph to 5 mph. Figure 4.10 (a) shows the accuracy obtained using only one event at a time from each trace, for both the datasets. For the subset of 16 drivers and for the mail van experiment, we notice that Ignition Switch Usage and Shift Usage are one of the first events captured by the OBD device. This is primarily because accessing information about Door Open and Door Close may not be possible in all vehicles before the ignition is turned on. Considering this limitation of some vehicles, we also evaluate our system using Ignition Switch Usage (ISU) as the origin event. Figure 4.10 (b) shows the accuracy for all traces with origin event set as ISU. *We observe that of all pre-trip fields, Release Brake is the most important field for rapid driver differentiation.*

We also choose Analysis of variance (ANOVA) method to investigate the important fields. In this method, we calculate the mean for each field (field mean). We then calculate the mean for all fields combined, the overall mean, followed by the standard deviation within a field, for each field. Finally, we calculate the standard deviation of each field mean from overall mean. Figure 4.11(a) shows the statistics for each field that we get using ANOVA method. From this Figure, we observe that the highest between-group variations come from SF and ISU events.

Next we explored the importance of the driving fields in our mailvan dataset. The driving fields used are BP, AP, RPM, TP, TS, V and SWA. Figure 4.11(b) shows the accuracy achieved using each driving field alone for driver distinction. It must be noted that these events occur long after a person has started driving, and hence may not be as useful for applications that can use this data early on before the start of the trip. *We observe that BP and SWA are the most important driving fields in distinguishing*



(a) Mailvan experiment

(b) Controlled experiment

Figure 4.8: Confusion matrix for driver differentiation.

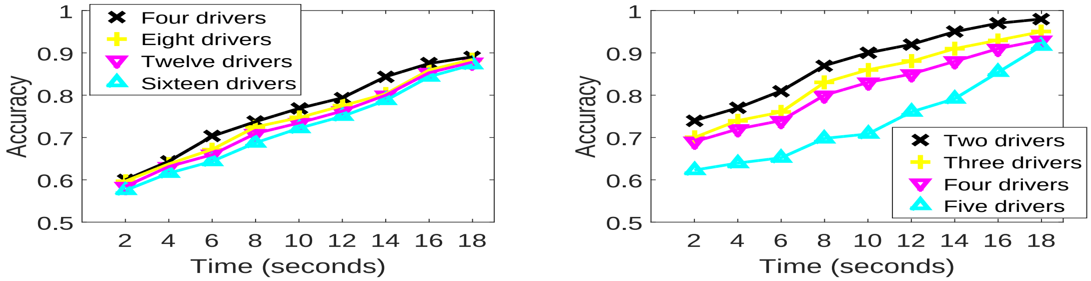
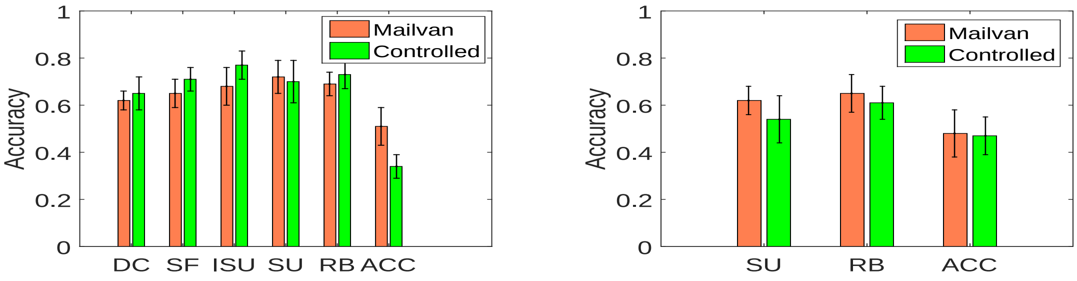


Figure 4.9: Varying number of drivers (a) Controlled experiment, (b) Mailvan experiment

drivers, and provide the highest accuracy when used alone.

Effect of training size

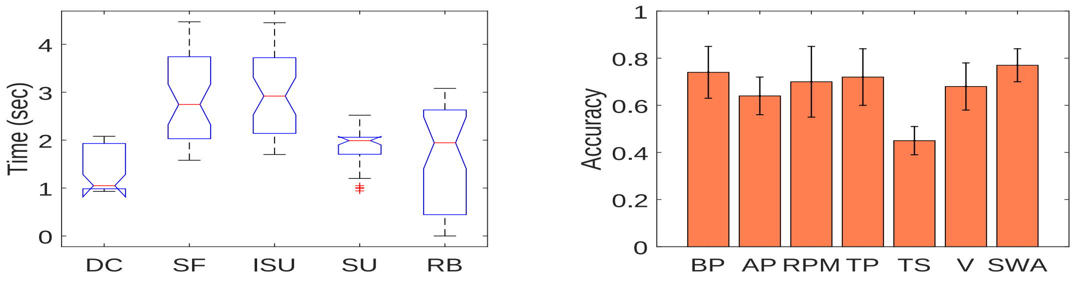
For supervised learning using 2-step driver validation, choosing the portion of data to be used as a training set is critical since the remaining data is used for testing the algorithm. If the training set is too small, it may not be enough to cover all characteristics of dataset and performance results may not be good as expected. Or if that set is too big, there may not be enough data to be tested and performance results could be lower than expected. In our experiments, we choose to train our classifier on different number of traces. We select, as our training size, 50%, 60%, 70%, 80% and 90% of the data. The remaining data is then used as the test set. Figure 4.12 shows the accuracy of our classifier for different training set sizes. *As the training size increases, our algorithm provides better results as expected, and by using* 90% *of dataset as training set, we can get* 91% *and* 89% *accuracy for the mailvan and controlled*



(a) Origin at DO event

(b) Origin at ISU event

Figure 4.10: Performance of individual pre-trip fields.



(a) ANOVA comparison

(b) Accuracy comparison

Figure 4.11: Individual field importance.

experiments, respectively.

Unsupervised driver differentiation

While our system performs well under supervised learning, we are also interested in quantifying its performance when prior driver data is not available for learning, i.e. unsupervised learning. We use K-means clustering as our first example of an unsupervised learning algorithm, where we only provide the number of expected drivers, but no advance information about driver behavior or learning traces. This algorithm then assigns each trace to a cluster, based on its distance from the cluster. Another unsupervised learning algorithm we investigate is hierarchical clustering algorithm. This algorithm does not require the number of drivers (clusters) or the event trace. It builds clusters iteratively with each input trace. We compare these two clustering approaches with the SVM algorithm. Figure 4.13 shows the comparison between supervised and

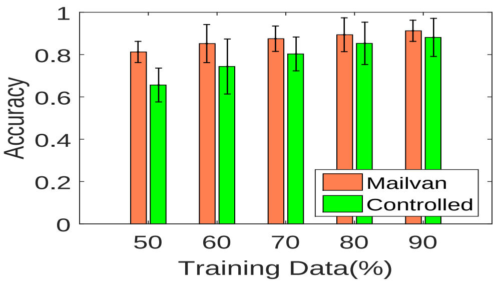


Figure 4.12: Accuracy over training data size.

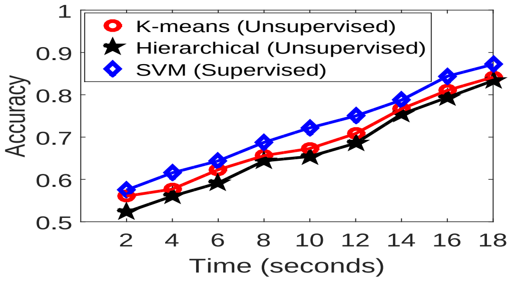


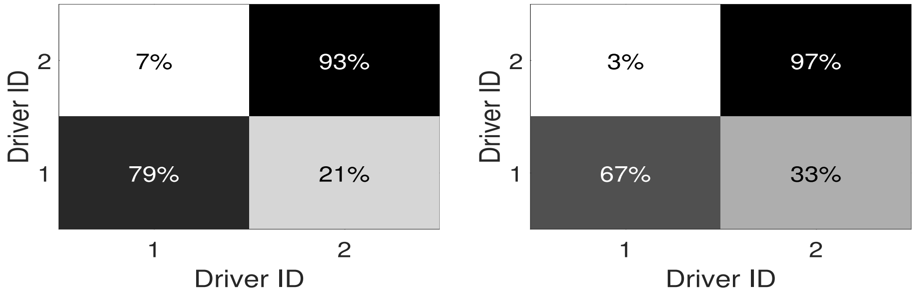
Figure 4.13: Performance of supervised vs unsupervised learning algorithms.

unsupervised approaches for driver classification. *It is evident that at an accuracy of* 89% *in less than 20 seconds, supervised learning performs only slightly better than the unsupervised learning approach, with* 84% *accuracy for the same trace length.*

Personal vehicle use setting

While sharing of a single vehicle is most common in commercial settings, members of a household may also share a car. In some cases the sharing is somewhat uniform, where the drivers spend similar amounts of time driving the car. In other scenarios, each car may have a primary driver and an occasional secondary driver. We hypothesize that pre-trip sequences within members of a household may undergo higher variation than professional drivers. They may encounter more distractions from the time they approach the car, to the time they start driving. Moreover, one might think behavioral similarities are higher among family members, specially when teenagers learn driving from their parents. To this end, we conducted a three-week user study with two different households, where members share a car for personal use.

Experiment setting. Household 1 is a married couple, and Household 2 is a



(a) Household 1

(b) Household 2

Figure 4.14: Confusion matrix for household experiments.

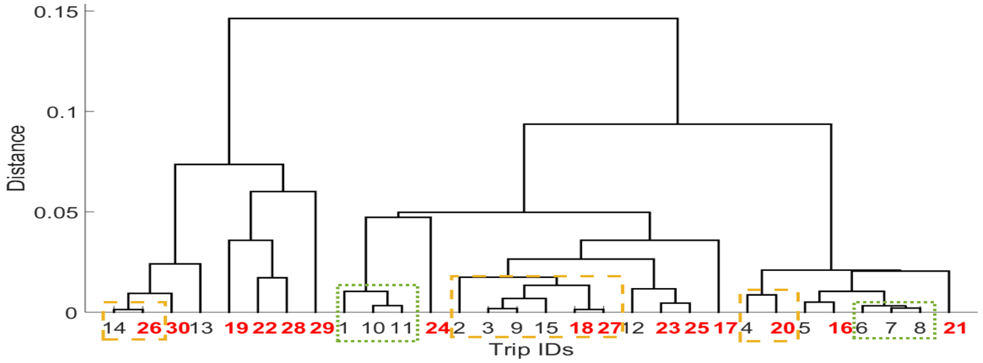


Figure 4.15: Similarity measure for 2 different drivers in a household. Trips IDs for husband: 1 to 15, Trip IDs for wife: 16 to 30

mother-daughter pair. Members of household 1 drove a Chevrolet Impala 2017, while the sensor data was being logged on a Nexus 4. About 3 days per week the wife drops the husband at work and uses the car for her daily chores. Most of her drives are non-routine and differ from day to day. Her stops are approximately one hour long. At the end of the day, she picks up the husband from work, and they go home together. On the other days, the husband drives himself to work and uses the car for lunch etc. They usually share driving tasks over the weekend. We collected 66 trips over a period of 3 weeks and logged all the pre-trips events and driving data on the smartphone. Each segment is considered a new trip if the engine was turned off. In the mother-daughter scenario, the mother, who is the primary driver, drives the car to work every day and back home. She drives a Chevrolet Equinox 2016, and a Nexus 5X was used for logging data. On some days she even drives it out for lunch, and other daily chores. The daughter drives the car far less, and thus is the secondary driver, often only over the weekends. We use this scenario for new driver detection, wherein we want to identify when the driver is not the primary driver of the vehicle. This entails what is known as one-class learning.

Evaluation. Fig 4.14(a) shows the confusion matrix for Household 1, where we use our adaptive outlier rejection algorithm with 5-fold cross validation, for distinguishing between husband and wife. The detection accuracy of this classifier was 85.6%. The data collected for the couple was significantly different from that observed during the commercial setting of the mailvan experiments. The start up sequences were longer by a factor of 10. During the exit interview, the couple informed that several times they made calls after entering the car, before starting to drive. Additionally, the order of pre-trip events was observed to be more irregular for personal use scenario, as compared to the professional use case.

Figure 4.15 depicts a dendrogram to visualize the similarity between drivers in Household 1. The vertical axis indicates the average distance between clusters, using correlation as the distance metric. Thus, lower distance implies higher correlation. The height of a node represents the distance of the two clusters that the node joins. We randomly select 15 trips from each driver. Trips IDs 1-15 are obtained from the husband, and Trip IDs 16-30 are obtained from the wife. The yellow dashed boxes mark trips from different drivers with very high correlation. Green boxes mark trips from the same driver.

For Household 2, where one of the driver is driving the car far less, we use clustering for one-class learning. We collected 34 trips from the primary driver and 3 trips from the secondary driver. The cluster is created using 31 random trips from the primary driver, and the remaining trips are used to calculate the distance from the centroid of the cluster. If the distance is above an empirical pre-calculated threshold, then that trip is registered as a new driver trip. Since each trip from the primary driver could have some outliers, we tried to cover all cases by using randomly selected trips to create the cluster. Averaging over 10 iterations, we obtain 73.3% accuracy in classifying test trips, as shown in Fig 4.14(b). *We observe that the correlation between drivers from a household who use a vehicle is higher compared to professional drivers, and the day to day variation much higher.*

4.5 Discussion

We have presented the design, implementation and evaluation of a driver differentiation technique using only pre-trip events. Unlike previous work, that focuses on parameters collected during driving, such as speed, most visited locations, etc., our work aims at sensing pre-trip events, that occur before the driver starts driving. Parameters monitored during the drive are a reflection of the driving style, but are greatly influenced by external driving conditions, such as traffic. Moreover, collection of values such as speed and most visited locations is detrimental to user privacy, as has been shown in previous work [33, 49]. Our proposed system focuses on sensing events that happen before a person starts driving. All these events are common acts that any driver conducts before a drive, such as closing the door, fastening the seatbelt, turning the ignition on etc. It is, however, the sequence and interval of these simple events that presents some inherent habits that differ widely from person to person. While this study has focused primarily on professional drivers, it is also an interesting question whether the results hold in family settings with multiple drivers. It is possible that random events of daily life lead to more variability in startup routines, which would make identification more challenging. It is also possible, however, that startup routines are more diverse across drivers within a family, because of greater differences in driving experience and trip purpose than in our professional drivers who all drove to deliver mail. The latter could lead to improved differentiation results.

In sensing these events, we employ the innocuous sensors that are already present on most vehicles these days. One might claim that adjusting mirrors, seat settings can also be used to differentiate drivers. Unfortunately, in our test vehicles, mirror positions are adjusted manually and seat positions could not be retrieved from the dongle. Therefore, these settings cannot be used to distinguish the drivers.

One may wonder how robust this technique is to larger variations in driver behavior. This can happen in situations where drivers break out of their pre-trip routine, for example, when they are interrupted by a phone call right after fastening the seatbelt. In such scenarios one would expect the system accuracy to be poor. However, we assume that some of these situations may have arisen in our real-world mail van study too, where our system achieved an accuracy of 90%. This indicates that our algorithm is at least somewhat robust to changes in driver behavior.

Depending on the application scenario, having an accuracy less than 100% may not be enough. But it must be noted that this 90% accuracy is achieved by using pre-trip data. For higher accuracy, we could always include additional driving and telemetric data such as vehicle speed, engine's RPM and steering wheel angle.

There are several potential privacy implications of this result. On one scenario, the result suggest that pre-trip data is potentially useful in re-identifying drivers in anonymous in-vehicle datasets. Storing pre-trip data together with anonymous streams of privacy sensitive data is therefore undesirable. It is worth noting, however, that re-identifying a driver in an anonymous dataset would require pre-trip profiles labeled with driver names, which are not always straightfoward to obtain. In another scenario, pre-trip data such as the timing of door closing events will likely be considered less privacy sensitive than driving data, which can be linked to visited locations, driving speeds, or aggressive driving styles. If the choice is between collecting driving data or pre-trip data for driver differentiation, pre-trip data therefore presents an opportunity to differentiate drivers using a less sensitive source of data.

4.6 Conclusion

We have shown that pre-trip in-vehicle data are particularly distinctive and represent a minimal set of in-vehicle data for driver differentiation. Driver differentiation based on pre-trip data requires only 20 seconds of data, while driving telemetry data approaches require about 10 minutes of data to reach a comparable accuracy. Specifically, we have shown that drivers can be distinguished using only pre-trip vehicle sensor data from the CAN bus with an accuracy of 91% in a real-world dataset of 480 trips collected over five mail vans with five drivers. In a controlled experiment, where all drivers steer the same vehicle along the same route, we have also shown that up to 16 drivers can be distinguished with similar accuracy. This accuracy can be further increased by combining it with driving telemetry data. It is also worth noting that the real-world mail van experiment was performed with mass-market vehicles that are close to 10 years old. With a larger number of sensors and electronic control systems in newer vehicles and luxury vehicles, one can expect even higher accuracy. The study has been focused on distinguishing drivers from the same vehicle. Whether a driver profile also holds across multiple vehicle models remains an open question. These findings have implications for the privacy treatment of such data and also enable novel in-car personalization, driver-specific insurance, targeted advertising, or unauthorized driver detection applications.

Chapter 5

COLLABORATIVE REAL-TIME TRAFFIC ESTIMATION

5.1 Overview of Chapter

With high computing power and less power constraints, vehicles provide plentiful opportunities to sense the dynamic environment. We propose to use sensing vehicles as edge compute nodes, focusing on sensing and interpretation of traffic from live video streams. Unlike smartphones, that are constrained in compute resources and available power, vehicles can support efficient compute platforms without the constraints of a small form factor compute node. Additionally, they provide wide reach into remote areas, where other platforms may be unavailable.

With the help of the front facing cameras that are installed in vehicles, we propose to record the traffic and count the vehicles that are traveling. Further, the average speed experienced by each driver can be used to estimate the traffic. Under free-flow traffic conditions drivers have the flexibility to choose higher speeds. However, when the density of vehicles increases, the vehicle speeds tend to decrease. Cameras allow capturing this information from many surrounding vehicles, and often many oncoming vehicles. They can therefore gather rich data about traffic conditions.

Existing work. Well-known navigation apps such as Google maps or Waze are using the GPS traces of some users that use these apps on road to calculate the speed of these vehicles and using that estimate the overall traffic conditions. However, since these apps are not being used by every single driver on road, the calculated speed is belong to the drivers that report GPS traces, potentially introducing systematic bias due to uneven sampling; for instance, the calculated speed could belong to a speeding driver in the left lane or a cautious, slow driver in the right lane. Therefore, the estimated traffic information may not be accurate. By using traffic cameras that are deployed on roads more complete traffic data could be obtained at one location. How many vehicles are traveling, what are their speeds, is there congestion on a given road are just a few questions that could be answered using traffic cameras and have been investigated previously by [81], [45], [69], [23] and [11]. However, the number of deployed traffic cameras is not sufficient to cover all roads and vehicles, especially in remote areas.

This work seeks to overcome these challenges with a collaborative sensing system, with vehicle detection, vehicle tracking and traffic estimation components, as shown in Figure 5.1, which uses a dash-cam mounted in vehicle and a video processor to process the images. The vehicle detection component of such a system could work continuously to detect vehicles and determine bounding-boxes around each vehicle. The vehicle tracking component then tracks the further movement of each detected vehicle. With the traffic estimation component, we can then count the number of vehicles on roads, estimate the lane in which they are traveling and their speeds using image processing techniques.

The salient contributions of this work are summarized below.

- An automated traffic estimation framework that manages vehicle detection, vehicle tracking and traffic estimation using a dashboard camera that can achieve wide coverage at low cost.

- We evaluate through multiple days of roadway experiments on a campus road, an interstate highway and a major highway, and show it is possible to count the vehicles that are traveling on a given road and determine their speeds. Our system can detect about 90% of vehicles traveling in the left most lane, estimate their speeds with about 10% error.

- Lane estimation for vehicles in the camera's view. Our system could achieve more than 91% lane estimation for the vehicles that are traveling in the left most lane, a.k.a passing lane.

5.2 Related Work

There has been much work about vehicle detection and tracking. For vehicle detection, most work [81], [45], [69], [23] and [11] assumes that the camera is static and vehicles are detected by finding the differences of the images for that camera. Using well-known background subtraction techniques, the only moving objects, vehicles, have been identified and speed estimations are made. In [81] and [45], the authors have attempted to calibrate the traffic camera using scene information for a particular camera and managed to count vehicles and estimate their speeds. [69] and [23] extended this idea and made it possible to cover any stable camera by first calculating the relative position of the traffic camera to vehicles, then estimating the lane boundaries and finally calculating the mean vehicle speed for each lane. Beymer et al. [15] proposed to use corner features to estimate the traffic flow. Hsu et al. [41] propose to use entropy to estimate vehicle speeds.

Cameras are not the only sensors to detect and track vehicles. Sonar and camera are being used at the same time in [47], which achieves vehicle detection at close distances (i.e., sonar distance). [17] shows that using multiple sensors provides better accuracy in object detection.

In computer vision, for object detection a set of robust features (SIFT [51], convolutional [26] etc.) from images is calculated and then classifiers are used to identify objects. These classifiers are run using a slide window on some parts of the image. This strategy has been used in many projects [76], [24], [75], [31].

The recent YOLO system [65] creates a single convolutional network that can detect multiple objects in an image. It needs a training period to work on full images and object coordinates that needs to be done only once, and then it can process the entire image, without a need for sliding window, and provide relatively accurate object detection almost in real-time (with latency of 25 ms).

In the realm of object tracking, Xiang et al. [78] proposed a multi object tracking framework based on Markov decision processes (MDP). They have two stages: first they collect ground truth trajectories of pedestrians, then a second learning method takes place as decision process is done by current status and history of the target. At every step, MDP attempts to track the target pedestrian and collects feedback from the ground truth. Then a similarity function is updated with the feedback. The authors manage to track pedestrians 7% better than the second best tracker comparison.

In another work [27], the authors proposed a framework to estimate trajectories of nearby vehicles using four cameras placed diagonally on the car. They modify the MDP tracker that's also being used in [78] to track vehicles. Since the movement of vehicles are not as random as pedestrians, the tracking performance is much better than in the previous effort. With 4 cameras, the trajectory recall is over 90%.

Lane estimation and tracking has been investigated in [52] and [43]. In this work, the camera, LIDAR and GPS sensors are used to extract road features such as lane markers and road curvatures, to enable applications such as a lane departure warning system and a driver attention monitoring system. However, only the traveling lane of the camera vehicle is estimated, not the other vehicles.

5.3 Background and Applications

With rising traffic congestion, many applications may benefit from an accurate estimation of traffic on a particular road. Let us consider the following examples.

Traffic Flow Terms. Traffic can be represented in several terms. Traffic flow represents the number of vehicles that are passing a reference point per unit time (e.g., vehicles per hour). Traffic density represents the number of vehicles per unit distance along the road. The higher either number is, the more congested the road becomes. As roads become more congested, vehicle speed decreases. There exists a well-known relationship between traffic congestion and vehicle speed.

For ease of interpretation, traffic congestion is often represented through a set of discrete Levels of Service (LOS). Table 5.1 [40] shows this quantization and the relationship between the foregoing parameters. In this table, level A represents free-flow traffic while level F represents congestion. The unit for the Traffic flow is vehicle/hour/lane and the unit for the density is vehicle/mile. Knowing the average speed of vehicles traveling on the road or the traffic-flow, LOS could be determined simply through a table lookup. By using the same table, it is possible to calculate how many vehicles are traveling per mile (i.e., traffic density on the road).

LOS	Speed Range	Flow Range	Density Range
A	Over 60	Under 700	Under 12
B	57-60	700-1100	12-20
C	54-57	1100-1550	20-30
D	46-54	1550-1850	30-42
E	30-46	1850-2000	42-67
F	Under 30	Over 2000	Over 67

Table 5.1: Levels of Service of a road

The traffic flow and density are direct metrics that shows the congestion of roads, or simply the traffic. With higher traffic flow and density numbers, one might expect busier traffic on roads.

Real-time Car Mapping. With the deployment of DSRC systems, vehicles have the capability to communicate with each other and learn the positions of nearby vehicles. However, for older cars that do not support DSRC, their location would not be known by other vehicles. Some newer vehicles come with built-in GPS receivers but since they don't transmit that information, the nearby vehicles do not have clue about the locations of these cars. Access to a fine-grained traffic estimation system creates awareness of a driver's surroundings by mapping cars in real-time.

Rear-End Collision Prevention. A rear-end collision is the most common traffic accident in the United States [1]. Vehicles traveling with a close proximity to the previous vehicle usually could not find enough time to stop if the vehicle in front needs to stop suddenly. With DSRC, these kinds of accidents would be expected to decrease because each vehicle supports DSRC, transmits their location and speed in real time. Being able to stop, of course, depends on the speed of other vehicles and the following distance. The earlier a stopped vehicle could be detected, the more time drivers have to stop on time. A stand-alone speed estimation system on each car, and does not rely on technology that may not be available on all vehicles, is the need of the hour. A continuous speed estimation of the vehicles around a car, can prevent many mishaps, such as rear-end collision prevention.

5.4 Vehicular Edge Nodes

Vehicles have evolved from mechanical systems to cyber physical systems, generating large amounts of real-time data. They offer high compute capabilities with far less power consumption concerns compared to other mobile platforms. Vehicles are powerhouses of energy, traversing through our physical world, and with the many sensors built in to them they are capable of sensing our dynamic environments.

Vehicles are increasingly being installed with front facing cameras. Originally, these cameras were intended for a specific purpose such as lane detection and lane keeping. However, recently, with the trend in autonomous driving, dashboard mounted cameras have been used for applications ranging from simple pedestrian/car detection [35, 34, 25, 14] to enabling a self-driving system [74, 59]. In addition to enabling vehicle specific or driver specific services, these cameras can be leveraged for large-scale traffic analytics. Cameras in each vehicle have a unique perspective of the observed environment. For example, a car driving in the left most lane has a clear view of the cars driving in the same direction, as well as those in the opposite direction. Similarly, a car in the rightmost lane is optimally placed for detecting stalled vehicles. Each car can be enabled to process raw video streams and compute high level semantic information. Pre-processing raw video streams to extract high level information optimizes bandwidth usage, reduces latency, and conserves privacy.

This information can then be shared with neighboring vehicles or a centrally located map service. The cloud-based map service can aggregate the information from a large number of vehicles to provide an up-to-date map of the region, overlaid with precise traffic and other road conditions information. This constitutes a more accurate and sophisticated assessment of regions, compared to other approaches such as surveillance cameras, that do not cover all areas. Such near real-time fine grained traffic analytics can enhance traffic flow and regulations, optimize transportation, and further assist in provisioning city services.

Using vehicles as edge compute platforms has become possible because of the increasingly powerful computing resources that are becoming available from multiple vendors.[1] These energy efficient compute platforms bring cutting-edge processors and accelerators to cars, enabling sophisticated and very deep networks to process rich video data in near real-time. As we bridge the gap in hardware, this work aims to demonstrate continuous large-scale traffic volume estimation techniques on distributed compute nodes, such as those in vehicles, to demonstrate that such compute resources are not just valuable for advanced driver assistance and automated driving systems but could also support a plethora of (potentially third-party) analytics applications if the platform becomes more openly programmable.

5.5 System Overview

Based on the aforementioned insights, we consider a traffic estimation system that seeks to detect and count vehicles, estimate the lanes they are traveling in and calculate how fast each vehicle is driven. The system consists of three main components: *vehicle detection*, *vehicle tracking* and *traffic estimation* as depicted in the Figure 5.1. In *vehicle detection*, the images collected by the dash camera is processed in-real time and a bounding box is created for each detected vehicle. With our parking lot experiments, we could observe up-to 6 vehicles could be detected in one frame. For real road tests, we could detect up-to 5 vehicles traveling in both directions. In *vehicle tracking*, we track the SIFT features of the vehicles to track them. Each vehicle is tracked individually. We do not focus on the full image, but the bounding box area since it is the part where the vehicle is detected. We calculate SIFT features for that area and by matching it for consecutive frames, we could track vehicles. To increase the stability, we focus on 5 consecutive frames, if the same vehicle is identified in 5 consecutive frames, we identify the car as the same vehicle. In *traffic estimation*, we first count the vehicles in the

---

[1] A well-publicized example is the NVIDIA DrivePX 2 [4] platform.

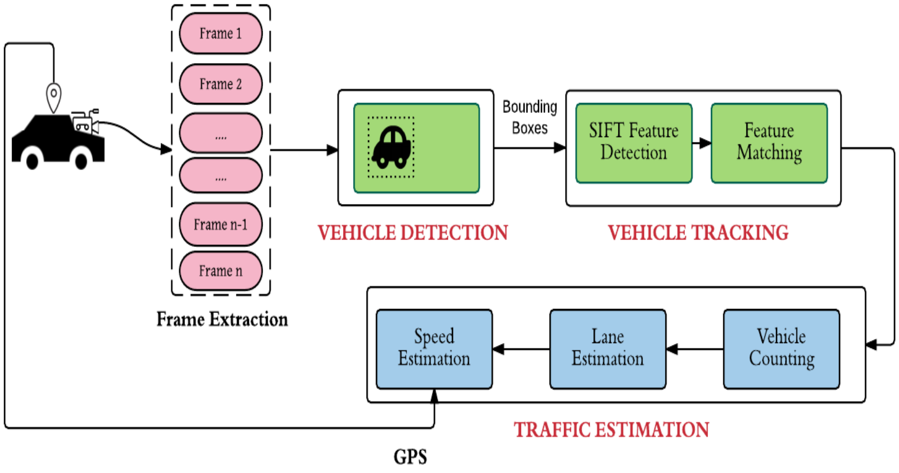


Figure 5.1: System Overview.

opposite lane by identifying unique cars. Then we decide in which lane that car is traveling by creating pseudo-lane markers. Finally, depending on the lane that car is traveling, we estimate the speed according to the Algorithm 4. Each component will be discussed in detail in the next subsection.

5.5.1 Approach

For each frame, *vehicle detection* component outputs bounding boxes around the detected vehicle(s). Since there could be multiple bounding boxes, we always use the left most bounding box of the left of the screen to identify the vehicle in the opposite lane. All vehicles detected in the right side of the screen are traveling in the same direction with the camera car.

Ideally, a vehicle traveling in the opposite direction would be detected by multiple consecutive frames. In order to get an accurate vehicle count on roads, we need to identify each unique vehicle. This can be achieved by finding similarity for the vehicles in consecutive frames. With the *vehicle tracking* component, we compare SIFT features of the vehicles for those frames. If we have a high number of matched feature points, those two vehicles should be the same one. After analyzing the frames, we observed that for the same vehicle in consecutive frames, there are minimum 15 matched features.

So we compare the feature points for different frames and if the number of matching features is greater than 15, those frames correspond to the same vehicle and we do not increase the count. Finally, we check for if the object is being identified 5 times. For some non-vehicle objects, our algorithm classifies them as vehicle. We exclude them by checking if we have minimum 15 matched feature for 5 consecutive frames. This algorithm is summarized in Algorithm 2.

One can claim that not all the feature points are coming from the vehicle, but also the outside world, such as road segment or trees. The feature point distinction is discussed in *traffic estimation*.

Algorithm 2 Vehicle Count Estimation

```
function $unique\_vehicle\_detection(f)$    ▷ record frames $f$ of vehicles traveling in the opposite direction
    for each consecutive frames $f_i$ and $f_{i+1}$ do
        $N=count\_sift\_features(f_i, f_{(i+1)})$;
                                                        ▷ Compare with the threshold
        if N ¿ 15 then
            $unique(i)=false$;
        else
            $unique(i)=true$;
        end if
    end for
end function
function INCREASE COUNT($unique$)    ▷ Increase the count if the unique vehicle is being seen minimum 5 times
    $count=count$ ++
end function
```

In *traffic estimation* component, we can first calculate the number of vehicles that are traveling by using the vehicle tracking results. For each unique vehicle, we increase the total count. The second step would be to estimate the lane of travel for each detected vehicle. For the vehicles that are traveling in the same direction, we propose to use Hough lines [3], to extract line segments based on Hough transform. With this way, we could detect the lines in the road and categorize each lane separately. For the opposite lane traveling vehicles, estimating the traveling lane is more tricky since we may not always observe the lane markers. We propose to create pseudo-lane markers in the opposite direction and estimate the traveling vehicle lane using those markers. The process is summarized in Algorithm 3.

In order to estimate the speed of each vehicle, we need to know how far that vehicle is moved for consecutive frames. We first calculate the distance change of the matched SIFT feature points in real world for consecutive frames. For this, the camera should be calibrated and this will be discussed in Section 5.6.4. With this way, we can calculate how each feature point is moved in real world as shown in *distanceCalculation* function in Algorithm 4. However, not all the feature points belong to the vehicle as can be seen in Figure 5.3. Therefore, some points may move differently for consecutive frames and the relative distance change for these points should be avoided to calculate the speed. This is the *detectDistanceAnomaly* function. And finally, by using the change of distance of the feature points of the vehicle, we can calculate the speed of that vehicle as shown in *Speed Estimation* function in Algorithm 4.

Our system does not require user interaction since it can automatically detect vehicles and estimate the speed of vehicles. The only exception is to estimate the lane for the opposite side traffic. For each road, we need the *Initialize* step in Algorithm 3 to create lane markers for that road, once.

5.6 Traffic Estimation

In this section, we describe how to detect vehicles on roads, calculate the speed and lane of travel information to estimate the traffic. While we discuss this in the context of vision, similar methods could be applied to other vehicle detection methods such as LIDAR.

5.6.1 Vehicle Detection

To detect vehicles on road in real-time, we need an object detection system that can detect vehicles regardless of make, model or color from any angle accurately and light-weighted such that it should support real-time detection. With that reason, we decide to use YOLO [65]: the object detection method using a single neural network. The network uses entire image features to predict objects and drawing bounding-boxes around them. With this method, the entire image is divided into grids and in each grid cells, predictions and confidence scores are generated for different objects. The higher the confidence score, the likelihood of correct object detection increases. YOLO is designed as a convolutional neural network: the initial layers are responsible to extract features and connected layers are responsible to predict the output objects with confidence scores. YOLO is extremely fast and streaming videos can be process with less than 25 ms of latency. Unlike classifier-based approaches, YOLO directly corresponds to detection performance and the entire model is trained jointly.

The training of this network is done with 20 different objects including dogs, horses, chairs, screens, benches, knives, books, beds, airplanes and cars. The authors use about 500 images for each object and input the locations of bounding boxes for each object in every image. When the network is trained, the output weight file is also provided. However, since YOLO is designed to identify wide range of objects, the car detection performance is not satisfying. With that reason we extend the provided car dataset with publicly available car datasets by Stanford University [8] and University of Illinois [9].

Training

With the extended dataset, we are using more than 8000 images of vehicles that are taken from every angle to train the network. The training has been done in a server with a GPU, Quadro K-5000 [2]. The training requires the images that includes the vehicles and the coordinates of the bounding boxes that cover the vehicles in each image. The training takes about 16 hours for 8000 images. This is a one-time process such that when the weight file is generated, we just need to run YOLO with that file on our test images. And YOLO can support real-time vehicle detection.

We also need to note that in the dataset, we are using the images of cars including sedans, SUVs, coupes, wagons with different colors and years. However, we don't include the images of trucks or buses. Therefore, the detection performance for those vehicles would not be as high as personal cars. If interested, with a dataset of those vehicles, YOLO could be trained as well.

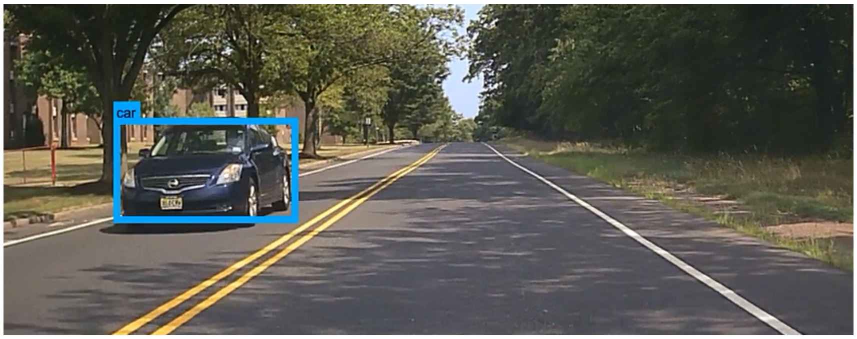


Figure 5.2: Detected vehicle.

Testing

For testing the accuracy of vehicle detection, we use a 30 fps camera and record a short footage in a parking lot with multiple vehicles. The primary objective is to detect vehicles in most of the frames. From our tests, we observed that the detection range is about 100 meters. For distances that are larger, the vehicle is not always detected. In Figure 5.2, a detected vehicle has been shown on the road. With extensive tests that were done both in parking lot and on roads, we observed vehicles are being detected regardless of model, color or type. In the night time, the performance of vehicle detection is not great, because in the training set, the vehicle images were taken in the day time. The detailed runs of testing will be discussed in Section 5.7.

5.6.2 Vehicle Counting

We consider a system that could count the number of vehicles traveling on roads in both directions. While traveling, the number of encountered vehicles in the same direction is usually not relevant since we either detect the same vehicles (e.g., our car is following them with constant speed or in a traffic jam) or we keep seeing new vehicles (e.g., we are passing vehicles) or don't see any vehicles (e.g., there is no car traveling in the detection range). However, for the opposite side, we encounter every single traveling car as long as the opposite side is visible. With that reason, we focus on counting the vehicles in the opposite direction.

YOLO can provide bounding-boxes around each detected vehicle. For one frame, counting the number of boxes could give an estimate about how many vehicles exist in that frame and it might work for a parking lot experiment. However, on real roads, every vehicle is moving as well as the camera so counting vehicles becomes more tricky. We need to avoid counting the same vehicle multiple times because that vehicle would be detected in several consecutive frames. To eliminate the counting the same car multiple times, we seek to determine if the vehicle detected in different frames is unique. By matching SIFT features of vehicles, we can determine if it's the same vehicle or a new one. Instead of calculating SIT features of the whole frame, we decide to use the left most bounding boxes for each frame. We first check if the left most bounding box is in the left side of the screen. If not, that vehicle is traveling in the same direction with us and we don't include it in the count. After making sure that vehicle is in deed traveling in the opposite direction, we compare the SIFT features of the objects in the bounding boxes for consecutive frame as discussed in Algorithm 2. If a vehicle is detected for 5 consecutive frames, we increase the count.

SIFT features of consecutive frames are calculated using VLFeat library [10] in Matlab. Sample matched features for 2 consecutive frames are shown in Figure 5.3. It should be noted that although some features do not belong to the vehicle, most of them do. In Algorithm 4, we discuss about how to eliminate features that don't belong to the vehicle.

We also need to note that vehicle counting depends on vehicle detection. For the vehicles that could not be detected for any reason, could not be counted as well. So the vehicles that are obscured by other vehicles, could not be counted with our algorithm.

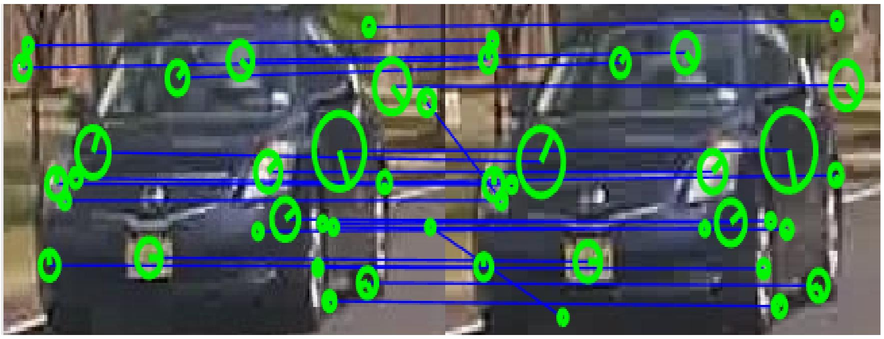


Figure 5.3: SIFT feature matching.

5.6.3 Lane Estimation

Intelligent vehicle systems enable applications that work with or for the human users with driver-assistance systems. Lane determination is an important concept of these apps and will also be used by autonomous driving. Lane keeping / departure warning systems have already been investigated in the literature and by tracking the lane markers, it's possible to determine the lane of travel for the camera vehicle. For the other vehicles traveling on roads, the similar study should be made.

With our system, by detecting and tracking the lane markers, we could detect the traveling lane for the vehicles that travel in our side. We propose the create lane regions using the lane markers and decide the traveling lane for each vehicle by intersecting the bounding box with the lane regions.

However, for the opposite direction, we can not always observe the lane markers due to barriers in between. In order to estimate the lane of which the vehicles that are traveling in the opposite direction, we propose to generate pseudo-lane markers that separate the lanes of travel. After creating the lane regions, we can repeat the process and decide on the traveling lane for the opposite lane traveling vehicles. The lane estimation results will be discussed in Section 5.7.

5.6.4 Speed Estimation

The average speed of vehicles determines the traveling time which is a good measure of road congestion and traffic performance. By knowing the speed of each traveling vehicle, an average speed of that route could be calculated and this info could be used by many application. By using stereo cameras, the depth information could be achieved and this could be used to calculate the speed of vehicles. However, the range of these cameras are usually in the order of 20 meters and this may not be the case for vehicles traveling in the highways. Instead, we propose to use a single dash-camera to estimate the speed of each vehicle. Since we don't know the distance of the car to our camera, we propose to calculate the distance traveled by that vehicle from one frame to another. If we assume the vehicle is traveling parallel to the camera, by using the camera parameters, we could estimate how far the vehicle is moved in real world during that frame. First, we need to calibrate the camera.

Camera Calibration

In MATLAB, we can get the intrinsic camera parameters by using the camera calibration app [7]. With this parameters, we can get the real-world distance on to objects in one frame, just by knowing the pixel coordinates of them in the frame. For this reason, we take 20 pictures of the chessboard from different angles and different distances. The other input of the app is the actual size of chessboard square (3 cm.). The squares of chessboard are detected by the app and shown in Figure 5.4. The output of this app is focal length of the camera (both in x and y axis) and intrinsic parameters that are needed to calculate the distance between two objects on the image.

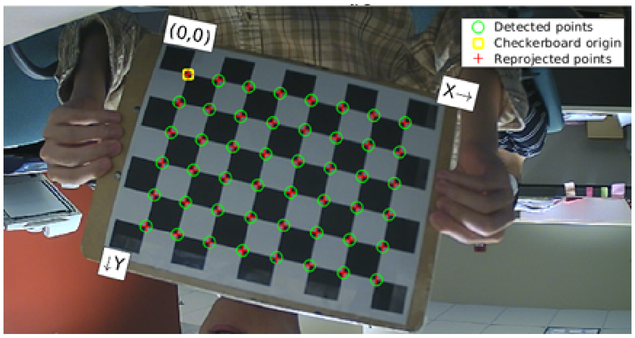


Figure 5.4: Detected squares on a chessboard.

After knowing these parameters, one can calculate the relative distance change of an object in real world, assuming that object moves parallel to the camera by knowing the horizontal distance between the object and the camera.

To check the calibration performance, we conducted several tests where multiple objects have been placed in a straight line and a picture is taken by the camera facing parallel to that line. The maximum distance error is less than 2% for both inside and outside experiments.

Te next step would be to find the speed of the object. After observing an object moves about x meters in consecutive frames, the speed of that object would become (30 * x) m/s for a 30 fps camera.

For the vehicle speed estimation, we can calculate how far each matched feature point moved from one frame to another as described in the Algorithm 4. However, not all the matching features belong to the vehicle and they might move differently. With that reason, we use the distance values that are in 1 sigma interval of the average distance value for consecutive frames. Since most of the feature points would belong to the car and move similarly, with this way we can eliminate the feature points of non-vehicle objects such as trees, lane markers etc.

One can claim that this speed estimation method works only for the straight roads, and when vehicles travel parallel to the camera vehicle. Although that's a valid argument, even if the road is not straight, vehicles move almost parallel to each other if we consider only a short period of time where two vehicles are close to each other. With this reason, we use last 10 frames of a vehicle in the view of our camera, to calculate the relative distance change. If one vehicle is tracked for more than 10 frames, we only use last 10 frames to estimate how far that vehicle is moved. As a final step, since we know how much time it took for the object to move that distance (by using the specs of the camera), we can calculate the speed of the object. In Section 5.7, we present the speed estimation results for three different routes we use.

5.7 Performance Evaluation

For evaluating our system, we aim to answer the following questions:

- What is the vehicle detection accuracy on roads?

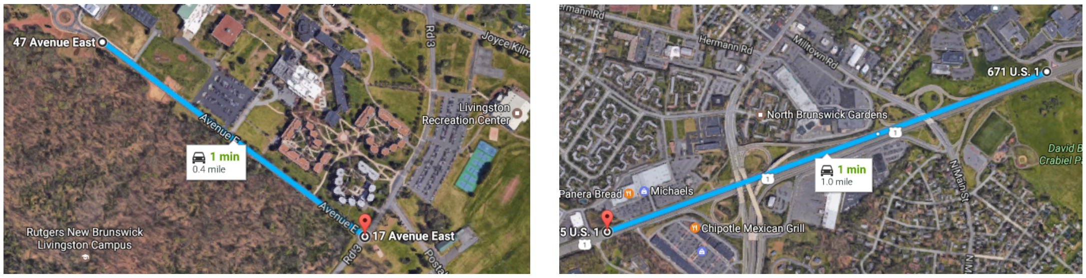
- How does this accuracy change with varying traffic conditions?
- Is it possible to detect lanes of the traveling vehicles?
- What is the speed estimation accuracy?
- How does this system perform compared to GPS-based traffic estimation systems?

To answer these questions, we conducted the following experiments.

5.7.1 Experimental Setup

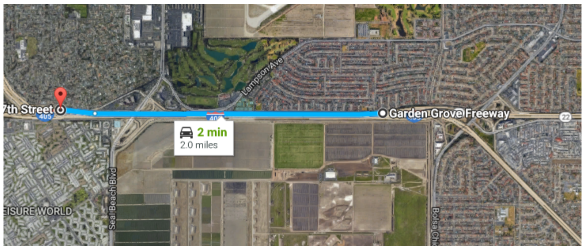
Hardware. In the proposed platform, we need a front facing camera that can work with different lighting conditions. After several real-world experiments, we observed that Go Pro camera performance deteriorates with exposure to direct sunlight. However, OmniVision camera gives similar performance results even with the direct exposure of sunlight. The second component is a graphical processing unit (GPU) to process the recorded stream in real-time. With a TX1 board, more than 30 fps could be achieved. The vehicle detection is achieved by a real-time object detection method. We use YOLO: Real-Time Object Detection method to detect vehicles. We conduct our experiments in three different routes in the United States. Route 1 is a campus road with two lanes, one in each direction. Route 2 is an interstate highway in New Jersey with 3 lanes in each direction. Route 3 is a major highway in California, with 8 lanes in each direction. In Route 1 and 2, we use 1 Omnivision OV 10635 camera that is placed on the windshield of a car. For Route 3, we use a GoPro Hero 5 camera. We use a server with a GPU, Quadro K-5000 to process the collected images.

Real Road Scenarios. Route 1 is a 0.4 mile stretch of a campus road, depicted in Figure 5.5(a). We use two vehicles traveling in opposite directions. We placed the camera in one vehicle and use GPS in both vehicles to calculate speed for each vehicle. Since there is only one lane in each direction, we focus on counting vehicles, and estimating the speed of each vehicle. We watch collected videos manually to obtain the ground truth for vehicle count, and use GPS-based speed values to get the speed estimation accuracy for our system.



(a) Route 1

(b) Route 2



(c) Route 3

Figure 5.5: Experiment roads.

Route 2 is a national highway through New Jersey, shown in Figure 5.5(b). We covered a distance of 1 mile on this highway during our experiments. We use total of 4 cars; three of them traveling in the same direction, one after the other and the fourth car with the camera traveling in the opposite direction. In all cars, GPS is enabled to calculate the speed as ground truth. We focus on counting vehicles in the opposite lane correctly and estimating the speeds of target vehicles with minimal error. We use the speed of known vehicles as ground truth for evaluation. Since there are multiple lanes, we also investigate the lane of travel for each traveling vehicles. For vehicle counting and lane estimation accuracy, we manually label the ground truth.

Route 3 was a distance of 2 miles covered on a major highway, as shown in Figure 5.5(c). In this experiment, we use total of 8 cars; 4 traveling in one direction and 4 in the other direction. Each vehicle is equipped with a GPS-enabled GoPro camera. Again, the vehicle counting and lane estimation labeling is performed manually. The ground truth for vehicle speeds is obtained through GPS.

Metrics. In this work, we evaluate the performance of vehicle detection using the following two metrics: (1) detection rate (DR) is the percentage of vehicles that are detected by our algorithm. For example, if there are x number of cars are traveling and if our algorithm can detect y of them, then the detection rate becomes (y/x). (2) False detection rate (FD) is the case when our algorithm detects a non-vehicle object as a vehicle.

Secondly, we evaluate the performance of our lane estimation technique by looking at the detection accuracy. For example, if a target car is traveling in the left most lane and if our algorithm is detected the lane of travel as the left, the accuracy is 1, otherwise 0. We evaluate the lane estimation for vehicles traveling in both directions.

Thirdly, we evaluate the performance of the speed estimation by calculating the error; the difference between the estimated and true speed of the vehicle. True speed of the vehicle is collected using GPS traces.

5.7.2 Experimental Results

Counting Accuracy. For Route 1, we drove our vehicle with the camera and detected all vehicles traveling in the opposite direction. Since there is only one lane in the opposite direction, all vehicles were detected and counted perfectly. We performed 10 driving tests on this route, and counted a total of 25 vehicles traveling in the other direction.

For Route 2, we drove our vehicle with the camera in the left most lane of the highway in order to observe more vehicles driving on the opposite side. With a total of 4 trips, we observed that all vehicles traveling in the left most lane could be detected, but some vehicles traveling in the middle and right most lane could not be detected because those cars are partially or fully obstructed by another vehicle. Table 5.2 shows the detection rate and false positive rate for 4 different driving experiments made on that highway. When the traffic is heavy, our detection rate suffers slightly, since more vehicles in the far away lane are now obstructed by vehicles in the left lane. We define any non-vehicle detection as a false positive. These are generated from the vehicle detection module. However, as mentioned in Section 5.6.2, we only consider a detection as a vehicle if it is detected in consecutive frames. Since false detections only appear intermittently, they are not counted as vehicles, since they are not detected in

Traffic Condition	LL-DR	ML-DR	RL-DR	Overall-DR	FPR
Light	94.7%	91.6%	90.4%	92.1%	2
Light	100%	88.4%	84%	91.5%	1
Heavy	95.5%	70.8%	62.8%	75%	0
Heavy	98.1%	76.9%	56.3%	78.1%	0

Table 5.2: Detection performance for Route 2

Car ID	LL-DR	ML-DR	RL-DR	Overall-DR	FR
1	88.8%	90.0%	73.6%	84.2%	0
2	94.4%	90.0%	78.9%	87.7%	0
3	94.4%	95%	89.4%	92.9%	0
4	100%	90%	84.2%	91.2%	0

Table 5.3: Detection performance for Route 3

consecutive frames.

For Route 3, we drove on two HOV lanes with two vehicles in each lane. All vehicles are equipped with GPS-enabled GoPro cameras. Vehicles 3 & 4 are traveling in the first HOV lane, 1 & 2 are traveling in the second HOV lane (i.e., second lane from the left). By this test, we compare the GoPro and Omnivision cameras as well. With direct exposure to sunlight, the distortion in GoPro cameras is high, and no vehicles were detected for a minute during the drive. After the sun's position changed, we could detect the vehicles. Our detection results are presented in Table 5.3. Regardless of the lane, the vehicle counting accuracy is comparable; we could detect at least 48 out of 57 vehicles. For first HOV lane, the detection rate is slightly higher because the cameras can capture more vehicles driving in the opposite direction.

Lane Estimation. To facilitate lane estimation, the system needs to first define the lane regions. To that end, we focus on detecting lane markers in the view of the camera. Figure 5.6 shows the detected left lane marker with a yellow line and the right lane marker with the purple line. The region in between is the traveling lane used by the camera vehicle. By using the features of the detected lines (i.e., the yellow solid

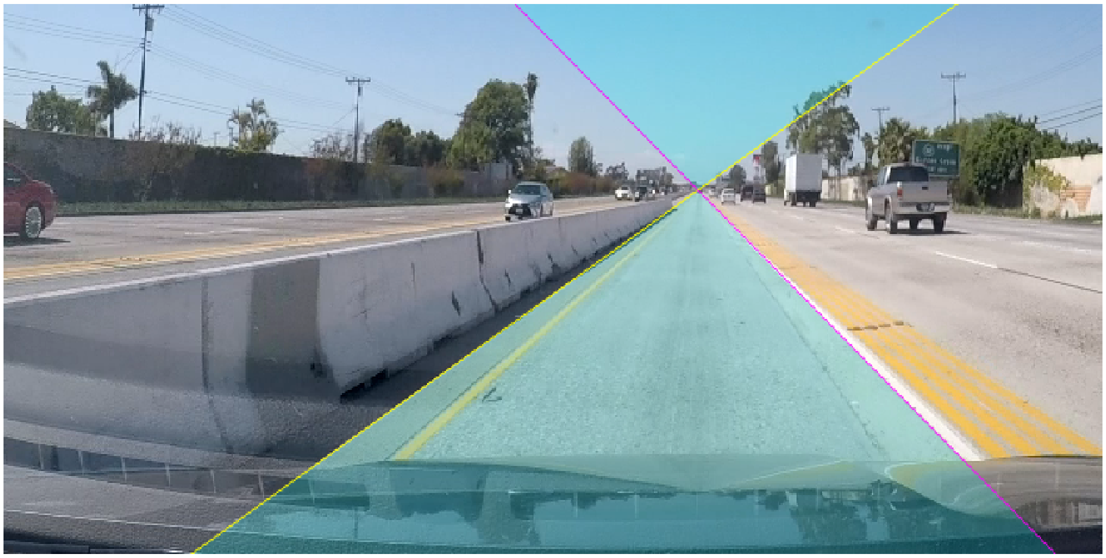


Figure 5.6: The lane of travel.

line, white solid line or white dashed line), we can also determine if the traveling lane is the left most lane, one of the middle lanes or the right most lane for multi-lane roads. Using lane markers on the same side of the road as the camera vehicle, we can generate lane regions. For each detected vehicle, the system determines the traveling lane by checking the overlap region of the vehicle's bounding box with the nearest lane region. The lane region with the maximum overlap is said to be the traveling lane for that vehicle.

Unfortunately, for the opposite direction, we can not always observe the lane markers due to barriers in between. In order to estimate the lane in which the vehicles are traveling in the opposite direction, we propose to generate pseudo-lane markers that separate the lanes of travel. As discussed in Algorithm 3 in Section 5.5, we create artificial lane markers and then classify the lanes using those markers. Figure 5.7 shows the number of vehicles per lane calculated by this algorithm for a sample road. Using our algorithm, we can estimate how many vehicles are traveling in each lane, in real time.

For ground truth, we manually count the number of vehicles for each lane and create a confusion matrix to evaluate the performance of this technique for Route 2 and Route

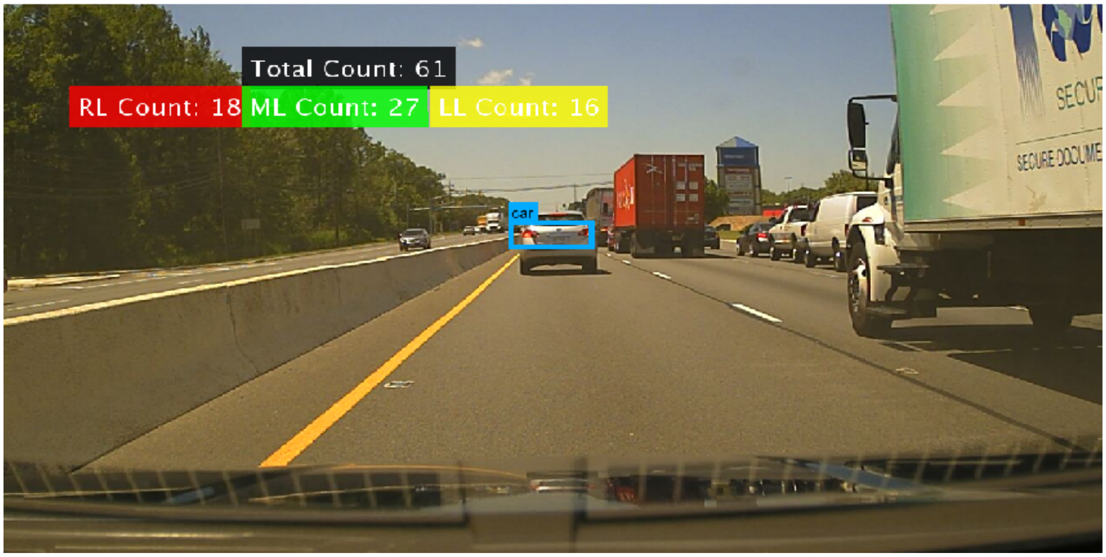
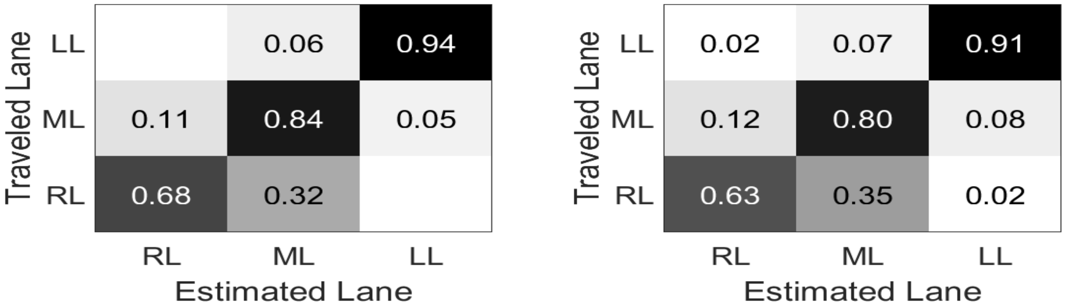


Figure 5.7: Vehicle count for each lane.



Traveled Lane \ Estimated Lane	RL	ML	LL
LL		0.06	0.94
ML	0.11	0.84	0.05
RL	0.68	0.32	

(a) Route 2

Traveled Lane \ Estimated Lane	RL	ML	LL
LL	0.02	0.07	0.91
ML	0.12	0.80	0.08
RL	0.63	0.35	0.02

(b) Route 3

Figure 5.8: Confusion matrix for roads

3, as shown in Figure 5.8. From this figure, we can observe that the estimated lane performance is highest for vehicles that are traveling in the left most lane. For the middle lane, we still have high estimation rate, however, almost one-third of the right lane traveling vehicles are identified as they are traveling in the middle lane. Also, on Route 2 the lane estimation accuracy is higher than Route 3. One possible reason is that on Route 3, traveling vehicles is a lot further than in Route 2 and this could cause problems about localizing each vehicle.

We want to emphasize that lane estimation needs a manual guidance that needs to be done for once, for each road. In this process, two random vehicles need to be tracked

Relative Speed	Estimated Speed	Error
50	50.5	1.01%
60	61.6	2.71%
70	73.4	4.85%
80	81.4	1.76%
90	93.1	3.4%
95	99.4	4.6%

Table 5.4: Speed estimation error for Route 1

for each lane and then pseudo-lane markers are created by using the trajectory of these vehicles and finally, lane estimation is achieved by comparing the center coordinates of each new vehicle with the lane markers.

Speed Estimation. As we discussed in Section 5.6.4, by using the intrinsic camera parameters and the horizontal distance between the camera and the target car, we can calculate the speed of the target car.

In Route 1, there is only one lane in each direction, therefore, the horizontal distance between the target car and the vehicle is 3.7 m, which is the average lane width in the United States. Both the target car and the camera car are driven at different speeds and encountered 6 times in Route 1. Table 5.4 shows the speed estimation results for this road. The first column represents the relative speed of two vehicles compared to each other. For example, if the target car is traveling at 25 mph and the camera car is traveling at 35 mph, the relative speed becomes 60 mph. Table 5.4 shows that the speed estimation works well for a variety of relative speeds and is always less than 5%.

In Route 2, since it's an interstate highway, we could drive with higher speeds and extend the results for speed estimation that we obtained from Route 1. In this test, all target cars are traveling in the left most lane of the opposite side at different speeds. We did the test with 3 vehicles driving in the opposite direction. The results are presented in Table 5.5. For Route 2, the error in the speed estimation is generally larger than for Route 1, but still less than 9%. One possible explanation could be there is a large

Relative Speed	Estimated Speed	Error
100	108.3	8.32%
110	116.7	6.01%
120	125.5	4.58%
130	139.2	7.07%
140	152	8.57%

Table 5.5: Speed estimation error for Route 2

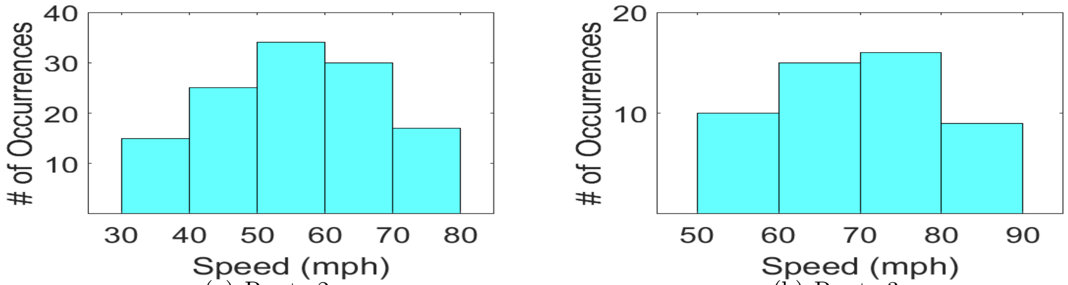
Relative Speed	Estimated Speed	Error
126	138.8	10.1%
127	142.1	11.9%
133	141.2	6.16%
135	149.7	10.9%

Table 5.6: Speed estimation error for Route 3

barrier in Route 2 that blocks the lower parts of the vehicles and causes fewer feature points to be detected, and be used for speed estimation. Also, the distance between target car and the camera car is also higher (6 m.) compared to Route 1 (3.7 m.).

In Route 3, the four camera vehicles travel in HOV-1 and HOV-2 (second from the left lane); two cars in each lane. Table 5.6 shows the estimated speed for the Route 3. The speed estimation error is less than 12% for all relative passes. Among three routes, this road has the biggest speed estimation error. One possible explanation is, we have about 27 m. between the left most lanes of each direction.

In Figure 5.9, we show the speed values used by drivers for two highways. In Route 2, vehicles are driven between 30-80 mph, in the majority of the drives, speed values 50-60 mph have been used. For Route 3, which is a major highway, the average speed experienced for each driver is higher, and for the majority of vehicles, speed values between 70-80 mph have been used. These results also prove that using GPS-based speed calculations for a few drivers could give very different results, so GPS-based traffic estimation may not be accurate.



(a) Route 2

(b) Route 3

Figure 5.9: Speed values for vehicles on roads

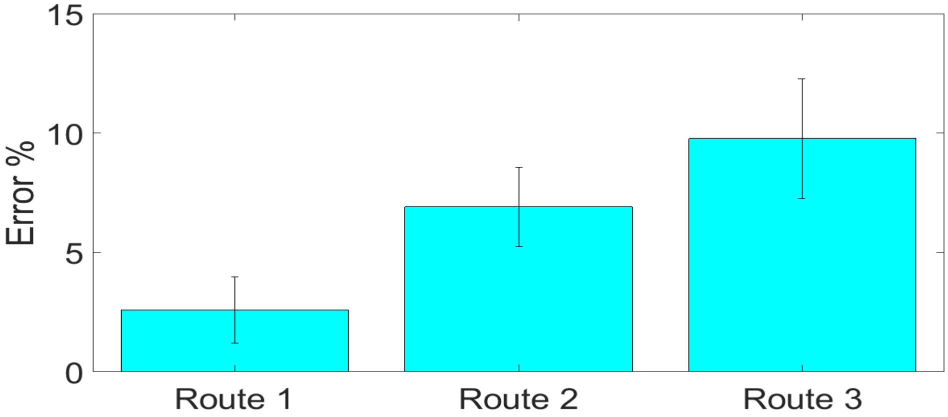


Figure 5.10: Speed estimation error for all roads.

In Figure 5.10, we show the error in the speed estimation for all three roads. For Route 1, which is a one-lane road in each direction, we have about 2.5% error in the speed estimation. For Route 2, the error is slightly higher, 6.9%. This is because, in Route 2 traffic for each direction is separated with a big barrier, and therefore some parts of the vehicles in the opposite direction are obscured. It affects our speed estimation because we are using feature matching between consecutive frames, and more matched feature points enable better estimation. In Route 3, we have about 9.7% error in the speed estimation. For this route, the gap between traveling lanes is even larger and we have even less matched feature points.

5.8 Discussion

We have presented the design, implementation and evaluation of traffic estimation technique using the live video streams collected within a moving vehicle. Unlike traffic surveillance camera based works, our work aims at detecting vehicles using front facing cameras in vehicles in real-time. When driving a vehicle in the left most lane, we have a clear view of the cars driving in both directions and it enables us to count the maximum number of cars on roads.

In estimating the speed of vehicles, we employ camera parameters and calculate the relative distance change of vehicles between frames assuming that vehicle is moving parallel to the camera. One might claim that roads are not always straight so this assumption might not be valid. However, we only focus on 10 frames before the vehicle leaves the camera view, and for this duration, the movement of the vehicle is almost parallel to the camera.

Our system could easily be used by enforcement forces. Since the speed estimation works with the error less than 12% even for the roads with wide barriers, speed monitoring could be managed when patrolling on the road. With this way, wide coverage could be achieved at a low cost compared to using speed radars or speed guns.

One might argue that GPS-based navigation systems can also calculate the speed so why such a vision-based system would be needed. With those systems, speed is being calculated by using a mobile device traveling on road with GPS activated. Due to battery problems, a lot of users do not activate GPS for a long period of time. Even if they do, those navigation systems can only calculate the speed experienced by that mobile phone which could easily give an outlier speed. For example, if the GPS is enabled for a motorcycle driver, the estimated speed would most likely be higher than the average speed of vehicles or if the GPS is enabled for an old driver, the estimated speed would be lower than the average speed of vehicles. On the other hand, by using our system, we could estimate the speed of each vehicle. So, we could estimate the speed experienced by each vehicle and calculate the average speed of vehicles on that road.

Depending on the application scenario, having a detection accuracy less than 100% may not be enough. Since we are using front facing cameras that are deployed in the vehicle, some vehicles traveling in the opposite direction, especially traveling in the right most lane, could be obstructed by other vehicles in between. Since the vehicle is partially observed by the camera, our object detection component may not identify the object as the vehicle. However, all vehicles that are traveling in the left most lane of the other direction would be identified as long as the barrier between the lanes is not high. By counting the number of vehicles traveling in the left most lane of the other direction, we can still decide on the traffic conditions. Also, by using a camera that's placed on the rood of the vehicle, the vehicle count accuracy could be increased.

5.9 Conclusion

In this work, we use vehicles as edge compute nodes and estimate the traffic from video streams using front facing cameras. With a real-time deep neural network object detection method, we can detect vehicles in both directions and count the number of vehicles traveling. This number could give much meaningful information about the traffic when combined with the estimated speeds of vehicles. Such information could be shared with neighboring vehicles or a map service. A service that collects such information from large numbers of vehicles could create an up-to-date map with fine-grained, near-realtime vehicle positions for that road. This results with more accurate and sophisticated assessment of roads, compared to traffic surveillance cameras, that do not cover all roads. With such a system, traffic flow can be enhanced and route planning for new cities could be achieved.

We used a dash-camera to collect the footage of the traffic and a GPU-equipped laptop for real-time vehicle detection and tracking. Specifically, we have shown that all oncoming vehicles could be detected and counted and tracked with less than 5% speed estimation error in two-lane campus road trials. In highway experiments, we achieved a minimum 75% vehicle counting accuracy under crowded traffic conditions and over 90% accuracy for light traffic conditions. The speed estimation error is about 9% for the interstate highway and less than 12% for the major multi-lane highway. This error can further be decreased for most common traffic roads, with shorter median strips. For example, we observed about 3% error in speed estimation for campus road experiments with no median strip on the road. We also have shown that the lane of travel for each vehicle could be estimated in both directions. Specifically, we could estimate the lane of travel perfectly for the vehicles that are traveling in the same direction with the camera vehicle and over 90% for the vehicles traveling in the left most lane of the opposite direction. Future work can integrate these techniques into a live traffic-view of real-time vehicle locations and speeds.

Algorithm 3 Traveling Lane Estimation

```
function INITIALIZE($r$) ▷ We manually record the trajectories of two vehicles from each lane and note down the center coordinates for each bounding box for road $r$ and calculate the line equations for each lane
    $ll\_flow\_line = a_1 \ . \ x + b_1$
    $ml\_flow\_line = a_2 \ . \ x + b_2$
    $rl\_flow\_line = a_3 \ . \ x + b_3$
end function
function PSEUDO LANE MARKER GENERATION($r$) ▷ for a particular road $r$, record frames $f$ of vehicles traveling in the opposite direction
    $Initialize(r)$ ▷ We calculate the lane markers
    $ll\_marker = (a_1{+}a_2)/2 \ . \ x + (b_1{+}b_2)/2$
    $rl\_marker = (a_2{+}a_3)/2 \ . \ x + (b_2{+}b_3)/2$
end function
function ESTIMATE THE LANE($bb$) ▷ Compare the center coordinates $coord$ of the bounding box $bb$ with the pseudo-lane markers
    if $coord$ ¿ $ll\_marker$ then
        $left\_lane\_vehicles{=}left\_lane\_vehicles$ ++
    else if $rl\_marker$ ¡ $coord$ ¡ $ll\_marker$ then
        $middle\_lane\_vehicles{=}middle\_lane\_vehicles$ ++
    else
        $right\_lane\_vehicles{=}right\_lane\_vehicles$ ++
    end if
end function
```

Algorithm 4 Vehicle Speed Estimation

```
function DISTANCECALCULATION($L, f$)    ▷ record frames $f$ of vehicles traveling in the opposite direction, $L$ is the horizontal distance between the vehicle and the camera
    for each consecutive frames $f_i$ and $f_{i+1}$ do
        $x_{i,j}, x_{i+1,j}$=$get\_sift\_features(f_i, f_{i+1})$;
                                                                ▷ Calculate the path distance
        for each SIFT feature $j$ do
            $d_j$=$calculate\_path\_distance(L, x_{i,j}, x_{i+1,j})$;
        end for
        $m(d)$=$average(d_j)$
        $\sigma(d)$=$std\_dev(d_j)$
    end for
end function
function DETECTDISTANCEANOMALY($m(d), \sigma(d), d$)
    for each SIFT feature $j$ do
        if $m(d) - \sigma(d) < d_j < m(d)+\sigma(d)$ then
            $valid_d$=$valid_d \cup d_j$
        end if
    end for
end function
function SPEED ESTIMATION($valid_d, fps$)
    $Speed$=$valid_d$x$fps$
end function
```

Chapter 6

Future Directions and Conclusions

This dissertation describes new applications to solve very possible connected vehicle problems. We proposed a system with three components to enhance vehicle dependability. We proposed an interference monitoring framework with roadside monitoring stations and mobile detectors that can achieve wide coverage at low cost. We presented a system that allows distinguishing drivers based on a minimal set of in-vehicle sensor data that was gathered from the CAN bus. Furthermore, we proposed another system that could estimate the traffic information of the road, such as the number of traveling vehicles, their speeds and traveling lanes using a single in-car camera. However, there are still some challenges that remain investigated for future research.

6.1 System limitations for future jammers

Our goal is to design a low-cost detection mechanism that can be mass deployed in roadways, and thus our jamming detection mechanisms are designed to identify existing off-the shelf GPS jammers that continuously emit interference. Future jammers that may emit the interference for a short period of time could still be caught by the mobile detectors (e.g smart phones) since they could still be reporting any outages or SNR anomalies even though the strength of the jamming signals change over time. With such reports, the path of the jammer car can be interpolated to cameras for identification. The probability that a jammer would be detected by a stationary monitoring point on a single pass is smaller but after a sufficient number of passes (say, for a commuter), detection would still be likely.

While not every individual user will be motivated to install our mobile detector app for jammer detection, we envision that this mechanism can still be deployed in police vehicles, public transit systems, delivery truck fleets, etc. It could also become part of future connected vehicle standards or be integrated into smartphone platforms with careful energy management.

6.2 Privacy Concerns for Driver Identity

Our system could differentiate drivers with over 90% accuracy in our experiments and this result suggests that pre-trip data is potentially useful in re-identifying drivers in anonymous in-vehicle datasets. Storing pre-trip data together with anonymous streams of privacy sensitive data is, therefore, undesirable. It is worth noting, however, that re-identifying a driver in an anonymous dataset would require pre-trip profiles labeled with driver names, which are not always straightforward to obtain. In another scenario, pre-trip data such as the timing of door closing events will likely be considered less privacy sensitive than driving data, which can be linked to visited locations, driving speeds, or aggressive driving styles. If the choice is between collecting driving data or pre-trip data for driver differentiation, pre-trip data, therefore, presents an opportunity to differentiate drivers using a less sensitive source of data.

6.3 Enhancing Traffic Data from Multiple Vehicle Views

With our proposed system, the locations of traveling vehicles, with their speeds could be estimated. By using this information, a real-time map could be generated and positions of the vehicles could be estimated throughout the trip (e.g. the locations of a traveling car could be estimated after a one minute trip). However, this information may not be accurate because vehicles could be changing speeds for external factors such as encountering an obstacle or just by changing lanes. If the information collected by a single car could be merged with another car's view, the map could be updated. For that purpose, the views of cars that are traveling in the same direction could be merged. However, if two cars are traveling in opposite directions, the full map of the road could be generated because each vehicle would be generating the map for the opposite lane and complete map could be achieved.

Hopefully, with the evolution of connected vehicles, the techniques suggested in this dissertations will prove useful and increase safety and minimize trip times for vehicles.

References

[1] Analyses of rear-end crashes and near-crashes. `https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/analyses20of20rear-end20crashes20and20near-crashes20dot20hs2081020846.pdf`. Accessed: 2017-03-18.

[2] CUDA quadro-k5000. `http://www.nvidia.com/object/quadro-k5000.html`. Accessed: 2017-03-17.

[3] Hough transform. `http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm`. Accessed: 2017-04-09.

[4] Nvidia drive px. `http://www.nvidia.com/object/drive-px.html`. Accessed: 2017-04-23.

[5] OnStar by GM. Accessed: 2015-11-10.

[6] The openxc platform. Accessed: 2015-12-5.

[7] Single camera calibration app. `https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html`. Accessed: 2017-03-18.

[8] Stanford university cars dataset. `http://ai.stanford.edu/~jkrause/cars/car_dataset.html`. Accessed: 2017-03-17.

[9] Uiuc image database for car detection. `https://cogcomp.cs.illinois.edu/Data/Car/`. Accessed: 2017-03-17.

[10] Vlfeat. `http://www.vlfeat.org/index.html`. Accessed: 2017-03-18.

[11] E. Bas, A. M. Tekalp, and F. S. Salman. Automatic vehicle counting from video for traffic flow analysis. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 392–397. Ieee, 2007.

[12] R. Bauernfeind et al. In-car jammer interference detection in automotive gnss receivers and localization by means of vehicular communication. In *FISTS, 2011 IEEE Forum on*, pages 376–381. IEEE, 2011.

[13] R. Bauernfeind et al. *Analysis, Detection and Mitigation of InCar GNSS Jammer Interference in Intelligent Transport Systems*. Deutsche Gesellschaft für Luft-und Raumfahrt-Lilienthal-Oberth eV, 2013.

[14] M. Bertozzi, A. Broggi, A. Fascioli, T. Graf, and M.-M. Meinecke. Pedestrian detection for driver assistance using multiresolution infrared vision. *IEEE transactions on vehicular technology*, 53(6):1666–1678, 2004.

[15] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 495–501. IEEE, 1997.

[16] J. Bhatti et al. Development and demonstration of a tdoa-based gnss interference signal localization system. In *PLANS, 2012 IEEE/ION*, pages 455–469. IEEE, 2012.

[17] L. Bruzzone, D. F. Prieto, and S. B. Serpico. A neural-statistical approach to multitemporal and multisource remote-sensing image classification. *IEEE Transactions on Geoscience and remote Sensing*, 37(3):1350–1359, 1999.

[18] S. Choi, J. Kim, D. Kwak, P. Angkititrakul, and J. H. Hansen. Analysis and classification of driver behavior using in-vehicle can-bus information. In *Biennial Workshop on DSP for In-Vehicle and Mobile Systems*, pages 17–19, 2007.

[19] Chronos Technology. *CTL3510 GPS Jammer Detector.* http://www.navtechgps.com/assets/1/7/CTL3510_DS.pdf.

[20] Chronos Technology. *CTL3520 GPS Jammer Detector & Locator.* https://www.ettus.com.

[21] T. Churches and P. Christen. Blind data linkage using n-gram similarity comparisons. In *Advances in Knowledge Discovery and Data Mining*, pages 121–126. Springer, 2004.

[22] T. Churches and P. Christen. Some methods for blindfolded record linkage. *BMC Medical Informatics and Decision Making*, 4(1):9, 2004.

[23] D. J. Dailey, F. W. Cathey, and S. Pumrin. An algorithm to estimate mean traffic speed using uncalibrated cameras. *IEEE Transactions on Intelligent Transportation Systems*, 1(2):98–107, 2000.

[24] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[25] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):743–761, 2012.

[26] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Icml*, volume 32, pages 647–655, 2014.

[27] J. V. Dueholm, M. S. Kristoffersen, R. K. Satzoda, T. B. Moeslund, and M. M. Trivedi. Trajectories and maneuvers of surrounding vehicles with panoramic camera arrays. *IEEE Transactions on Intelligent Vehicles*, 1(2):203–214, 2016.

[28] M. Enev, A. Takakuwa, K. Koscher, and T. Kohno. Automobile driver fingerprinting. *Proceedings on Privacy Enhancing Technologies*, 2016(1):34–50, 2016.

[29] Ettus Research. *USRP N210.* https://www.ettus.com.

[30] Ettus Research. *VERT900.* https://www.ettus.com.

[31] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[32] D. Fontanella et al. In-car gnss jammer localization using vehicular ad-hoc networks. 5/6 2013.

[33] X. Gao, B. Firner, S. Sugrim, V. Kaiser-Pendergrast, Y. Yang, and J. Lindqvist. Elastic pathing: Your speed is enough to track you. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14, pages 975–986, New York, NY, USA, 2014. ACM.

[34] D. Gavrila. Pedestrian detection from a moving vehicle. *Computer VisionECCV 2000*, pages 37–49, 2000.

[35] D. M. Gavrila and S. Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *International journal of computer vision*, 73(1):41–59, 2007.

[36] GNU Radio. http://www.gnuradio.org.

[37] P. Golle and K. Partridge. On the anonymity of home/work location pairs. In *Pervasive computing*, pages 390–397. Springer, 2009.

[38] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Preserving privacy in gps traces via density-aware path cloaking. *Proceedings of CCS07*, 2007.

[39] B. Hoh, T. Iwuchukwu, Q. Jacobson, D. Work, A. Bayen, R. Herring, J.-C. Herrera, M. Gruteser, M. Annavaram, and J. Ban. Enhancing privacy and accuracy in probe vehicle-based traffic monitoring via virtual trip lines. *Mobile Computing, IEEE Transactions on*, 11(5):849–864, May 2012.

[40] W. S. Homburger and J. H. Kell. Fundamentals in traffic engineering. 1988.

[41] W.-L. Hsu, H.-Y. Liao, B.-S. Jeng, and K.-C. Fan. Real-time traffic parameter extraction using entropy. *IEE Proceedings-Vision, Image and Signal Processing*, 151(3):194–202, 2004.

[42] http://ngsim-community.org/. *Next Generation SIMulation Community*, 2013.

[43] A. S. Huang. Lane estimation for autonomous vehicles using vision and lidar. 2010.

[44] Y. Jiang, H. Qiu, M. McCartney, W. G. J. Halfond, F. Bai, D. Grimm, and R. Govindan. Carlog: A platform for flexible and efficient automotive sensing. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, SenSys '14, pages 221–235, New York, NY, USA, 2014. ACM.

[45] Y.-K. Jung and Y.-S. Ho. Traffic parameter extraction using video-based vehicle tracking. In *Intelligent Transportation Systems, 1999. Proceedings. 1999 IEEE/IEEJ/JSAI International Conference on*, pages 764–769. IEEE, 1999.

[46] C. Karatas, L. Liu, H. Li, J. Liu, Y. Wang, J. Yang, Y. Chen, M. Gruteser, and R. Martin. Leveraging wearables for steering and driver tracking,. In *IEEE International Conference on Computer Communications (Infocom) 2016*. ACM, 2016.

[47] S. Kim, S.-Y. Oh, J. Kang, Y. Ryu, K. Kim, S.-C. Park, and K. Park. Front and rear vehicle detection and tracking in the day and night times using vision and sonar sensor fusion. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2173–2178. IEEE, 2005.

[48] I. Kramer et al. Android gps jammer localizer application based on c/n0 measurements and pedestrian dead reckoning. In *ION-GNSS*, 2012.

[49] J. Krumm. Inference attacks on location tracks. In *Proceedings of the 5th International Conference on Pervasive Computing*, PERVASIVE'07, pages 127–143, Berlin, Heidelberg, 2007. Springer-Verlag.

[50] J. Lindstrom et al. Gnss interference detection and localization using a network of low-cost front-end modules. In *ION GNSS*, 2007.

[51] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[52] J. C. McCall and M. M. Trivedi. Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. *IEEE transactions on intelligent transportation systems*, 7(1):20–37, 2006.

[53] R. Mitch et al. Civilian gps jammer signal tracking and geolocation. In *ION GNSS*, 2012.

[54] C. Miyajima, Y. Nishiwaki, K. Ozawa, T. Wakita, K. Itou, K. Takeda, and F. Itakura. Driver modeling based on driving behavior and its evaluation in driver identification. *Proceedings of the IEEE*, 95(2):427–437, 2007.

[55] MobileMark Antenna Solutions. *GPS Multiband Antenna.*

[56] P. Mohan, V. N. Padmanabhan, and R. Ramjee. Nericell: Rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 323–336, New York, NY, USA, 2008. ACM.

[57] M. A. Mutaz et al. Leveraging platoon dispersion for sybil detection in vehicular networks. In *PST*. IEEE, 2013.

[58] T. Nighswander et al. Gps software attacks. In *CCS*. ACM, 2012.

[59] T. Okano. Method for correcting pixel data in a self-luminous display panel driving system, Feb. 15 2000. US Patent 6,025,818.

[60] X. Ouyang et al. Short-time fourier transform receiver for nonstationary interference excision in direct sequence spread spectrum communications. *Signal Processing, IEEE Transactions on*, 49(4), 2001.

[61] J. D. Parsons and P. J. D. Parsons. *The mobile radio propagation channel.* J. Wiley, 2000.

[62] A. Pentland and A. Lin. Modeling and prediction of human behavior. *Neural Computation*, 11:229–242, 1995.

[63] J.-P. Poncelet et al. A low-cost monitoring station for detection & localization of interference in gps l1 band. In *NAVITEC*. IEEE, 2012.

[64] G. D. Rash. Gps jamming in a laboratory environment. *NAWCWPNS*, pages 1–20, 1997.

[65] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[66] A. Riener and A. Ferscha. Supporting implicit human-to-vehicle interaction: Driver identification from sitting postures. In *The First Annual International Symposium on Vehicular Computing Systems (ISVCS 2008)*, page 10, 2008.

[67] M. Scannapieco, I. Figotin, E. Bertino, and A. K. Elmagarmid. Privacy preserving schema and data matching. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 653–664. ACM, 2007.

[68] R. Schnell, T. Bachteler, and J. Reiher. Privacy-preserving record linkage using bloom filters. *BMC medical informatics and decision making*, 9(1):41, 2009.

[69] T. N. Schoepflin and D. J. Dailey. Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. *IEEE Transactions on Intelligent Transportation Systems*, 4(2):90–98, 2003.

[70] L. Scott. J 911: Fast jammer detection and location using cell-phone crowdsourcings. *GPS World*, 2010.

[71] E. N. Skomal et al. *Measuring the radio frequency environment.* Van Nostrand Reinhold Company, 1985.

[72] H. Summala. Automatization, automation, and modeling of driver's behavior. In *Recherche - Transports - Scurit*, pages 35–45. Elsevier, 2000.

[73] U-Blox. *EVK-7N.* https://www.u-blox.com/en/evaluation-tools-a-software/gps-evaluation-kits.html.

[74] C. Urmson et al. Self-driving cars and the urban challenge. *IEEE Intelligent Systems*, 23(2), 2008.

[75] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 606–613. IEEE, 2009.

[76] P. Viola and M. J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[77] Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, and R. P. Martin. Sensing vehicle dynamics for determining driver phone use. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services, (MobiSys'13)*. ACM, 2013.

[78] Y. Xiang, A. Alahi, and S. Savarese. Learning to track: Online multi-object tracking by decision making. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4705–4713, 2015.

[79] J. Yang, S. Sidhom, G. Chandrasekaran, T. Vu, H. Liu, N. Cecan, Y. Chen, M. Gruteser, and R. P. Martin. Detecting driver phone use leveraging car speakers. In *Proceedings of the 17th annual international conference on Mobile computing and networking (Mobicom'11)*, pages 97–108. ACM, 2011.

[80] B. Zan, P. Hao, M. Gruteser, and X. Ban. Vtl zone-aware path cloaking algorithm. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 1525–1530, Oct 2011.

[81] Z. Zhu, B. Yang, G. Xu, and D. Shi. A real-time vision system for automatic traffic monitoring based on 2d spatio-temporal images. In *Applications of Computer Vision, 1996. WACV'96., Proceedings 3rd IEEE Workshop on*, pages 162–167. IEEE, 1996.

Appendix A

Refereed Publications as a Ph.D. Candidate

Hang Qiu, Favad Ahmad, Ramesh Govindan, Marco Gruteser, **Gorkem Kar**. Augmented Vehicular Reality: Enabling Extended Vision for Future Vehicles. In ACM HotMobile, 2017.

Jian Liu, Yan Wang, **Gorkem Kar**, Yingying Chen, Jie Yang, and Marco Gruteser. Snooping Keystrokes with mm-level Audio Ranging on a Single Phone. In ACM MobiCom, 2015.

Gorkem Kar, Hossen Mustafa, Yan Wang, Yingying Chen, Wenyuan Xu, Marco Gruteser, and Tam Vu. Detection of on-road Vehicles Emanating GPS Interference. In ACM CCS, 2014.

Appendix B

Acquired Patents as a Ph.D. Candidate

Gorkem Kar, Marco Gruteser, Fan Bai. Method and Apparatus of Differentiating Drivers Based on Driving Behaviors. U.S. Patent 62/257,565, 2015.